GENERATION OF SYMBOLIC FUNCTIONS VIA ORDERED BINARY TREE DATA STRUCTURE

TSE Yiu Keung

A thesis presented to The Chinese University of Hong Kong in partial fulfilment of the Degree of Master of Philosophy in Department of Electronics.

Department of Electronics,
The Chinese University of Hong Kong.

May, 1984.

GENERATION OF SYMBOLIC FUNCTIONS VIA ORDERED BINARY TREE DATA STRUCTURE

Abstract

A new computer oriented method for generating symbolic network functions is presented. A symbolic expression is represented by an ordered binary tree data structure. Each node of the binary tree represents one symbolic term and the symbolic expression is equal to the sum of the symbolic terms represented by all the nodes. The nodes of the binary tree are divided into different levels; the symbolic terms with same number of symbols are considered to be in the same level. By defining arithematic operations on the tree structure, we can apply arithematic operations on symbolic expressions. Therefore a symbolic network function can be generated just by direct evaluation of a symbolic determinant. The symbolic manipulation method may be used together with other algorithms to evaluate a symbolic determinant, such as Gaussmann Algebra or Signal Flow Graph method. The particular advantage of the method is that the cancellation problem of repeated symbolic terms is solved in the same time. Operations of tree structures can be easily implemented with the recursive programming technique on a digital computer system.

A program is written for implementing the new scheme. Through several examples, the new method is illustrated and compared with two other existing techniques. The result shows that with the aid of the efficient symbolic manipulation method, direct evaluation of symbolic determinant to generate the symbolic network function becomes straight forward.

Acknowledgement

I wish to thank Professor Chen Chih Fan and Dr. Yeung Kai Shing, who act as my supervisors; their guidance and suggestions are most valuable. Also I am very grateful to Croucher Foundation for granting me the studentship.

Contents

Chapter 1. Introduction

Generating symbolic network functions on a digital computer is a very attractive topic in the field of network analysis, because we can obtain much more information about a network from its symbolic network function than the corresponding numerical values. In other words, insight is easier to obtain, sensitivties are easier to be studied, and statistical analysis is much reduced [1]. For example to design a circuit, if we want to keep some parameters to be varied, the symbolic network function is extremely powerful and highly useful.

There are many techniques for studying network functions by use of numerical methods; however generation of symbolic network functions is much more difficult. Three existing methods for symbolic network functions which have been actually implemented and documented are as follows: [2]

1. Signal-flow-graph method [3],
2. Tree-enumeration method [4],

and 3. Parameter-extraction method [5].

The first two methods are classified as topological methods while the last one is considered as a numerical approach. Topological methods are those techniques which derive the network function from the structure of some graphs associated with the system. Signal Flow Graph method generates network functions by finding the loop weights of all the first order loops and the disjoint higher order loops of the signal flow graph formulating the networks, while the Tree Enumeration method generates network functions by summing up all the directed tree admittance products of the directed graph formulating the networks. A

serious drawback of topological methods is that the extremely large number of higher order loops or trees are generated for a network of moderate size. Therefore the methods are limited to small networks of about 15 nodes and 30 branches. By using topological methods, the computer time and storage grow rapidly as the size of the network increases. Also the required computer resources mainly depend on the sturcture of the network rather than the number of symbolic parameters in the network. It implies that the topological methods are not efficient to generate partially symbolic network functions. For some large circuits with only a few circuit elements represented by symbols, the numerical approach, Parameter Extraction Method is usually applied.

An important problem still remains unsolved. If a network consists of repeated symbolic elements, repeated symbolic terms will occur in the network function and the cancellation of those repeated symbolic terms becomes a very important problem in symbolic network analysis. However the symbolic manipulatiuon for cancelling the repeated symbolic terms is extremely difficult to be dealed with a digital computer.

In this thesis, we are going to present a new method for generating symbolic network functions which is based on expanding an symbolic determinant. The particular advantage is that the method can handle the problem of cancellation of repeated symbolic terms efficiently. It uses the hierarchical data structure or ordered binary tree data structure to represent symbolic expression. By defining arithematic operations on the hierarchical data structure, we can evaluate a symbolic determinant directly. In addition to generation of symbolic network functions, the expansion of symbolic determinant can be applied to evaluate the

transfer function of a state space model, which can be expressed as fraction of two determinants [8][9]. Also by expanding a symbolic determinant, we can find the inverse of a matrix with symbolic entries.

In chapter 2, the closed system philosophy and related technique for generating symbolic network function is described. In general, the closed system idea can be applied to most of symbolic network analysis algorithms. Also we can extend the technique for the case of multi-variable system analysis.

In chapter 3, two most widely used methods for generating symbolic network functions are briefly reviewed. The first method is the Signal Flow Graph Method which is a topological approach. And the second is the Parameter Extraction Method which is a numerical one.

In chapter 4, the new method is presented which is a symbolic manipulation approach using an ordered binary tree data structure to represent a symbolic expression. Applying arithematic operations on the tree structures, we can generate the symbolic network function by direct evaluation of a symbolic determinant. The method can be applied together with some other algorithms to evaluate a symbolic determinant. The Gaussmann Algebra and Signal Flow Graph method are used for showing how the repeated symbolic terms are cancelled efficiently when the new symbolic manipulation method is applied.

In chapter 5, several typical examples are illustrated for comparing the existing methods with the newly established method.

Chapter 2. Closed System

A closed system is defined as follows: the independent source of the system is changed into a controlled source just by adding a directed branch with weight Q from the output node to the input node; thus the system becomes independent of any external input. Then all the symbolic terms of the expansion of the system determinant can be sorted into numerator and denominator according to the existance of the additional weight, Q in each symbolic term.

Single input and single output closed system analysis [10]

For a linear time-invariant network with only one independent source, we can simply express the equilibrium equations for the network in frequency domain as follow:

$$A X = b U \tag{2.1}$$

where A is an n x n matrix,

U is the indepentent source,

b is an n x 1 vector of constant, and

X is an n x 1 vector for the unknown variables; they are currents for loop analysis, voltages for nodal analysis and a combination of voltages and currents for mixed analysis.

In detailed expression, A, b and X are given as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix}, \text{ and } \quad X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_n \end{bmatrix}$$

If X_i is the desired output, we can solve X_i in terms of U by applying Cramer's rule as follows:

$$X_i = \frac{b_1 \Delta_{1i} + \cdots\cdots + b_n \Delta_{ni}}{\Delta} U \tag{2.2}$$

where Δ is determinant of A,

and Δ_{ij} is the cofactor of a_{ij}.

Thus the network function becomes

$$T = \frac{X_i}{U} = \frac{b_1 \Delta_{1i} + \cdots\cdots + b_n \Delta_{ni}}{\Delta} = \frac{N}{D}. \tag{2.3}$$

To find the numerator N, we need to evaluate many symbolic determinants Δ_{1i}, Δ_{2i}, ..., and Δ_{ni}, and then add them together. In order to

simplify the work, it is desirable to generate the symbolic network function through the closed system formulation. The method is simply based on changing the independent source into a dependent source which is controlled by the output. Thus the system becomes independent of any input. By evaluating the system determinant and then sorting the symbolic terms, we can obtain the numerator and denominator seperately.

Consider the system equations Eq.(2.1) again, we change the independent source U into a dependent source which is controlled linearly by the output X_i such that

$$U = Q\, X_i \tag{2.4}$$

where Q is a dummy symbolic variable.

The closed system then becomes

$$A\, X = b\, U = b\, Q\, X_i \tag{2.5}$$

Regrouping the variable X_i, we have $A' X = 0$, or

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & (a_{1i} - b_1 Q) & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & (a_{2i} - b_2 Q) & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ a_{n1} & a_{n2} & \cdots & (a_{ni} - b_n Q) & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_n \end{bmatrix} = 0 \tag{2.6}$$

We denote the system determinant in Eq.(2.6) as Δ_{cs} which is equal

to the determinant of A'. After expanding Δ_{cs}, we have

$$\Delta_{cs} = \begin{vmatrix} a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2i} & \cdots & a_{2n} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ a_{n1} & \cdots & a_{ni} & \cdots & a_{nn} \end{vmatrix} + \begin{vmatrix} a_{11} & \cdots & (-b_1 Q) & \cdots & a_{1n} \\ a_{21} & \cdots & (-b_2 Q) & \cdots & a_{2n} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ a_{n1} & \cdots & (-b_n Q) & \cdots & a_{nn} \end{vmatrix} \qquad (2.7)$$

We then expand the second determinant at i-th column in cofactor form and take out the common factor Q, we have

$$\Delta_{cs} = \Delta - Q\,(b_1 \Delta_{1i} + b_2 \Delta_{2i} + \cdots\cdots + b_n \Delta_{ni}) \qquad (2.8)$$

Comparing Eq.(2.8) and Eq.(2.3), we get

$$\Delta_{cs} = D - Q\,N \qquad (2.9)$$

Since there is no system input, the system performance becomes independent of external input and we have $\Delta_{cs} = 0$. If we let $Q = \frac{1}{T}$, then $\Delta_{cs} = D - (\frac{1}{T})\,N = 0$. Therefore, we find that the network function: $T = \frac{N}{D}$. In Eq.(2.9), the numerator N accompanies with a negative sign. In order to eliminate it, we let $U = -P X_i$ which

means Q = -P, of course then $\Delta_{cs} = D + P\,N$. Thus we can generate the network function by expanding only one symbolic determinant Δ_{cs}; then sort all the symbolic terms into denominator and numerator according to the existance of the symbol P in each symbolic term.

Example 2.1

Consider the simple network shown in Fig. 2.2.

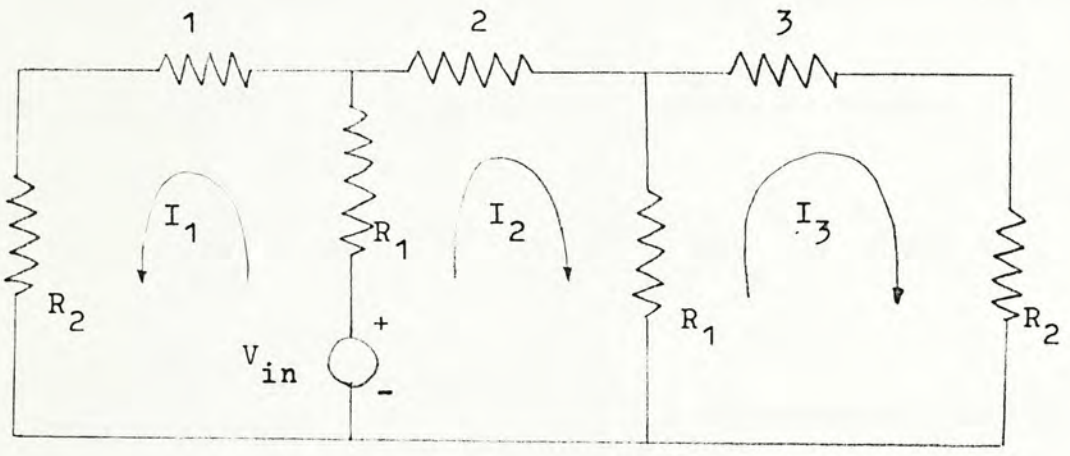


Fig. 2.2 : A simple network.

By applying loop analysis, we have

$$\begin{bmatrix} 1 + R_1 + R_2 & R_1 & 0 \\ R_1 & 2 + 2R_1 & -R_1 \\ 0 & -R_1 & 3 + R_1 + R_2 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} V_{in} \qquad (2.10)$$

Using the closed system technique, we first change the input to a dependent source related by the output I_3 as follows:

$$V_{in} = - P\, I_3 \qquad (2.11)$$

Substituting Eq.(2.11) into Eq.(2.10) and then regrouping the variables, we obtain

$$\begin{bmatrix} 1 + R_1 + R_2 & R_1 & P \\ R_1 & 2 + 2R_1 & P - R_1 \\ 0 & -R_1 & 3 + R_1 + R_2 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} \\ 0 \\ \\ \end{bmatrix} \qquad (2.12)$$

By expanding the closed system determinant Δ_{cs} and then sorting all the terms according to the existance of the symbol P, we have

$$\Delta_{cs} = P(R_1 + R_1 R_2) + 6 + 14R_1 + 6R_1^2 + 8R_1^2 + 12R_1^2 R_2 + 2R_2 + 2R_1 R_2 + R_1^2 + 2R_1^2 R_2 \qquad (2.13)$$

Seperating Eq.(2.13) into numerator and denominator, we obtain the following result:

$$\frac{I_3}{V_{in}} = \frac{N}{D} = \frac{R_1 + R_1 R_2}{6 + 14R_1 + 6R_1^2 + 8R_1^2 + 12R_1^2 R_2 + 2R_2 + 2R_1 R_2 + R_1^2 + 2R_1^2 R_2} \qquad (2.14)$$

We can easily and clearly obtain the network function by evaluating the closed system's determinant only.

Extension of closed system to multivariable system analysis

In the previous discussion, the closed system technique is only applied to the single input and single output system. The technique can be extended to multivariable system as well.

There is no loss of generality, we only concentrate our discussion on the two-input, two-output system. The same procedure can be applied to the systems with n-input and m-output, of course.

Consider the multivariable system shown in the following figure.

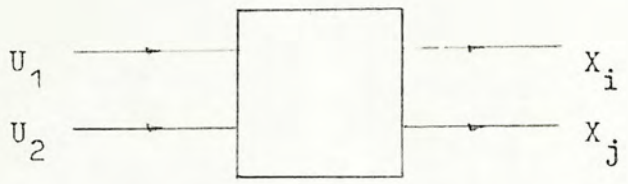


Fig. 2.1 : A two input and two output system.

The system can be described by the system equation

$$A\,X = b\,U \tag{2.15}$$

where A, b, X and U are defined as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad b = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ b_{n1} & b_{n2} \end{bmatrix}, \quad X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_n \end{bmatrix}$$

and $$U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$$

If X_i and X_j are the desired outputs, we can solve the outputs in terms of the inputs by use of superposition theorem [18] and Cramer's rule as follows:

$$X_i = \frac{b_{11}\Delta_{1i} + b_{21}\Delta_{2i} + \ldots + b_{n1}\Delta_{ni}}{\Delta} U_1 + \frac{b_{12}\Delta_{1i} + \ldots + b_{n2}\Delta_{ni}}{\Delta} U_2$$

$$X_i = \frac{N_{1i}}{\Delta} + \frac{N_{21}}{\Delta}, \tag{2.16}$$

and

$$X_j = \frac{b_{11}\Delta_{1j} + b_{21}\Delta_{2j} + \ldots + b_{n1}\Delta_{nj}}{\Delta} U_1 + \frac{b_{12}\Delta_{1j} + \ldots + b_{n2}\Delta_{nj}}{\Delta} U_2$$

$$X_j = \frac{N_{1j}}{\Delta} + \frac{N_{2j}}{\Delta}. \tag{2.17}$$

The transfer matrix relating the output vector and input vector can be shown as:

$$\begin{bmatrix} X_i \\ X_j \end{bmatrix} = \begin{bmatrix} \frac{N_{1i}}{\Delta} & \frac{N_{2i}}{\Delta} \\ \frac{N_{1j}}{\Delta} & \frac{N_{2j}}{\Delta} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{2.18}$$

Now, we change the independent sources U_1 and U_2 into dependent sources which are controlled by the outputs such that

$$U_1 = -(P_{1i} X_i + P_{1j} X_j) ,$$

and

$$U_2 = -(P_{2i} X_i + P_{2j} X_j) \qquad (2.19)$$

where the symbol P_{hk} relates the input U_h to the output X_k .

Substituting Eq.(2.19) into Eq.(2.15) yields

$$A X = -\begin{bmatrix} b_{11} \\ b_{21} \\ \cdot \\ \cdot \\ \cdot \\ b_{n1} \end{bmatrix} (P_{1i} X_i + P_{1j} X_j) - \begin{bmatrix} b_{12} \\ b_{22} \\ \cdot \\ \cdot \\ \cdot \\ b_{n2} \end{bmatrix} (P_{2i} X_i + P_{2j} X_j)$$

$$= -\begin{bmatrix} b_{11} P_{1i} + b_{12} P_{2i} \\ b_{21} P_{1i} + b_{22} P_{2i} \\ \cdot \\ \cdot \\ \cdot \\ b_{n1} P_{1i} + b_{n2} P_{2i} \end{bmatrix} X_i - \begin{bmatrix} b_{11} P_{1j} + b_{12} P_{2j} \\ b_{21} P_{1j} + b_{22} P_{2j} \\ \cdot \\ \cdot \\ \cdot \\ b_{n1} P_{1j} + b_{n2} P_{2j} \end{bmatrix} X_j \qquad (2.20)$$

Regrouping the variables X_i and X_j gives

$$\begin{bmatrix} a_{11} \cdots (a_{1i} + b_{11}\,P_{1i} + b_{12}\,P_{2i}) \cdots (a_{1j} + b_{11}\,P_{1j} + b_{12}\,P_{2j}) \cdots a_{1n} \\ a_{21} \cdots (a_{2i} + b_{21}\,P_{1i} + b_{22}\,P_{2i}) \cdots (a_{2j} + b_{21}\,P_{1j} + b_{22}\,P_{2j}) \cdots a_{2n} \\ \cdot \qquad\qquad \cdot \qquad\qquad\qquad \cdot \qquad\qquad \cdot \\ \cdot \qquad\qquad \cdot \qquad\qquad\qquad \cdot \qquad\qquad \cdot \\ \cdot \qquad\qquad \cdot \qquad\qquad\qquad \cdot \qquad\qquad \cdot \\ a_{n1} \cdots (a_{ni} + b_{n1}\,P_{1i} + b_{n2}\,P_{2i}) \cdots (a_{nj} + b_{n1}\,P_{1j} + b_{n2}\,P_{2j}) \cdots a_{nn} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_n \end{bmatrix} = 0 \qquad (2.21)$$

By expanding the system determinant at the i-th column and j-th column, the system determinant is equal to the sum of nine determinants:

$$\Delta_{cs} = \begin{vmatrix} a_{11} \cdots a_{1i} \cdots a_{1j} \cdots a_{1n} \\ a_{21} \cdots a_{2i} \cdots a_{2j} \cdots a_{2n} \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ a_{n1} \cdots a_{ni} \cdots a_{nj} \cdots a_{nn} \end{vmatrix} + \begin{vmatrix} a_{11} \cdots a_{1i} \cdots b_{11}\,P_{1j} \cdots a_{1n} \\ a_{21} \cdots a_{2i} \cdots b_{21}\,P_{1j} \cdots a_{2n} \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ a_{n1} \cdots a_{ni} \cdots b_{n1}\,P_{1j} \cdots a_{nn} \end{vmatrix}$$

$$+ \begin{vmatrix} a_{11} \cdots a_{1i} \cdots b_{12}\,P_{2j} \cdots a_{1n} \\ a_{21} \cdots a_{2i} \cdots b_{22}\,P_{2j} \cdots a_{2n} \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ a_{n1} \cdots a_{ni} \cdots b_{n2}\,P_{2j} \cdots a_{nn} \end{vmatrix} + \begin{vmatrix} a_{11} \cdots b_{11}\,P_{1i} \cdots a_{1j} \cdots a_{1n} \\ a_{21} \cdots b_{21}\,P_{1i} \cdots a_{2j} \cdots a_{2n} \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ a_{n1} \cdots b_{n1}\,P_{1i} \cdots a_{nj} \cdots a_{nn} \end{vmatrix}$$

$$
\begin{vmatrix}
a & \dots & b & P & \dots & b & P & \dots & a \\
11 & & 11 & 1i & & 11 & 1j & & 1n \\
a & \dots & b & P & \dots & b & P & \dots & a \\
21 & & 21 & 1i & & 21 & 1j & & 2n \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
a & \dots & b & P & \dots & b & P & \dots & a \\
n1 & & n1 & 1i & & n1 & 1j & & nn
\end{vmatrix}
+
\begin{vmatrix}
a & \dots & b & P & \dots & b & P & \dots & a \\
11 & & 11 & 1i & & 12 & 2j & & 1n \\
a & \dots & b & P & \dots & b & P & \dots & a \\
21 & & 21 & 1i & & 22 & 2j & & 2n \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
a & \dots & b & P & \dots & b & P & \dots & a \\
n1 & & n1 & 1i & & n2 & 2j & & nn
\end{vmatrix}
$$

$$
\begin{vmatrix}
a & \dots & b & P & \dots & a & \dots & a \\
11 & & 12 & 2i & & 1j & & 1n \\
a & \dots & b & P & \dots & a & \dots & a \\
21 & & 22 & 2i & & 2j & & 2n \\
\cdot & & \cdot & & & \cdot & & \cdot \\
\cdot & & \cdot & & & \cdot & & \cdot \\
\cdot & & \cdot & & & \cdot & & \cdot \\
a & \dots & b & P & \dots & a & \dots & a \\
n1 & & n2 & 2i & & nj & & nn
\end{vmatrix}
+
\begin{vmatrix}
a & \dots & b & P & \dots & b & P & \dots & a \\
11 & & 12 & 2i & & 11 & 1j & & 1n \\
a & \dots & b & P & \dots & b & P & \dots & a \\
21 & & 22 & 2i & & 21 & 1j & & 2n \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
a & \dots & b & P & \dots & b & P & \dots & a \\
n1 & & n2 & 2i & & n1 & 1j & & nn
\end{vmatrix}
$$

$$
+
\begin{vmatrix}
a & \dots & b & P & \dots & b & P & \dots & a \\
11 & & 12 & 2i & & 12 & 2j & & 1n \\
a & \dots & b & P & \dots & b & P & \dots & a \\
21 & & 22 & 2i & & 22 & 2j & & 2n \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
\cdot & & \cdot & & & \cdot & & & \cdot \\
a & \dots & b & P & \dots & b & P & \dots & a \\
n1 & & n2 & 2i & & n2 & 2j & & nn
\end{vmatrix}
\qquad (2.22)
$$

Extracting all the controlled source symbols from the nine determinants, we have

$$\Delta_{cs} = SEXP + P_{1i}(SEXP_{1i}) + P_{2i}(SEXP_{2i}) + P_{1j}(SEXP_{1j}) + P_{2j}(SEXP_{2j})$$
$$+ P_{1i}P_{1j}(SEXP_{1i,1j}) + P_{1i}P_{2j}(SEXP_{1i,2j}) + P_{2i}P_{1j}(SEXP_{2i,1j})$$
$$+ P_{2i}P_{2j}(SEXP_{2i,2j}) \qquad (2.23)$$

where $SEXP_{mn}$ is the symbolic expression with the controlled source symbol P_{mn}, and $SEXP_{mn,hk}$ is the symbolic expression with the controlled source symbols P_{mn} and P_{hk}.

Comparing Eq.(2.22) with Eq.(2.16) and Eq.(2.17), we find that

$$\Delta = SEXP$$

$$N_{1i} = SEXP_{1i}$$

$$N_{1j} = SEXP_{1j}$$

$$N_{2i} = SEXP_{2i}$$

$$N_{2j} = SEXP_{2j} \qquad (2.24)$$

Therefore we can obtain the whole transfer matrix by only evaluating one symbolic determinant; however, we have to do some redundant work which is due to the evaluation of the symbolic terms with more than one controlled source symbols, such terms are not directly related to the transfer matrix.

Example 2.2

A multi-variable system which is taken from [17], has the following governing equation:

$$\begin{bmatrix} s+2 & -1 & -1 \\ -1 & s+1+k & -k \\ -1 & -k & s+1+k \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{2.25}$$

The desired outputs are X_1 and X_2. We now change the input into controlled source related to the output as follows:

$$\begin{aligned} U_1 &= -(P_{11} X_1 + P_{12} X_2) \\ U_2 &= -(P_{21} X_1 + P_{22} X_2) \end{aligned} \tag{2.26}$$

Substituting Eq.(2.26) into Eq.(2.25), we have

$$\begin{bmatrix} s+2 & -1 & -1 \\ -1 & s+1+k & -k \\ -1 & -k & s+1+k \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -(P_{11} X_1 + P_{12} X_2) \\ -(P_{21} X_1 + P_{22} X_2) \end{bmatrix}$$

$$= \begin{bmatrix} -P_{11} & -P_{12} \\ 0 & 0 \\ -P_{21} & -P_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{2.27}$$

Regrouping the variables X_1 and X_2 yields

$$\begin{bmatrix} P_{11} + s + 2 & P_{12} - 1 & -1 \\ -1 & s + 1 + k & -k \\ P_{21} - 1 & P_{22} - k & s + 1 + k \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = 0 \qquad (2.28)$$

By expanding the closed system determinant, Δ_{cs} and sorting the terms according to the existance of the symbols of the dependent sources, we have

$$\begin{aligned} \Delta_{cs} = {} & (3 s + 4 s^2 + s^3 + 6 k s + 2 s^2 k) \\ & + P_{11} (1 + 2 s + s^2 + 2 k + 2 s k) + P_{12} (1 + s + 2 k) \\ & + P_{21} (1 + s + 2 k) + P_{22} (1 + 2 k + s k) \\ & + P_{11} P_{22} (k) + P_{12} P_{21} (- k) \\ & + P_{11} P_{12} (0) + P_{21} P_{22} (0) \end{aligned} \qquad (2.29)$$

Comparing Eq.(2.29) to Eq.(2.23) and Eq.(2.24), we have

$$\begin{aligned} \Delta_{cs} &= 3 s + 4 s^2 + s^3 + 6 k s + 2 s^2 k \\ N_{11} &= 1 + 2 s + s^2 + 2 k + 2 s k \\ N_{12} &= 1 + s + 2 k \\ N_{21} &= 1 + S + 2 k \\ N_{22} &= 1 + 2 k + s k \end{aligned} \qquad (2.30)$$

The symbolic terms with two controlled source elements $P_{11}P_{22}$, $P_{12}P_{21}$, $P_{11}P_{12}$, and $P_{21}P_{22}$ are the redundant terms which will not appear in the required transfer matrix.

Chapter 3. Review of Two Existing Algorithms

In this chapter, a brief review of two widely used methods for generation of symbolic network functions is made. The first method is the well known Signal Flow Graph; the second is the Parameter Extraction. Afer having the basic concepts about these two existing methods discussed, we will present the new method in the next chapter.

Signal Flow Graph Method

The well known signal flow graph [3] is a weighted directed graph representing a system of simultaneous linear equations according to the following rules:

1. Node weights (node variable) represent variables.
2. Branch weights (branch transmittances) represent the coefficients related to node weights.
3. Node variable = $\sum$ (incoming branch transmittances) $\times$ (node variable from which the incoming branch originates)

Definitions:

1. Source node : a node with only outgoing branches.
2. Dependent node : a node with some incoming branches.
3. Sink node : a dependent node with only incoming branches.

Now consider the reverse process of constructing a signal flow graph to represent a set of simultaneous equations. We start from the set of equations as follows:

$$\underset{nx1}{\underline{X}} = \underset{nxn}{\underline{\underline{A}}}\ \underset{nx1}{\underline{X}} + \underset{nxm}{\underline{\underline{B}}}\ \underset{mx1}{\underline{X}_s} \qquad (3.1)$$

where

X - dependent nodes.

X_s - source nodes.

A, B - branch transmittances,

and $a_{i,j}$ - transmittance of branch directed from X_j to X_i,

$b_{i,j}$ - transmittance of branch directed from X_{sj} to X_i.

From Eq.(3.1), we can solve for X in terms of X_s to obtain

$$X = \left[1-A\right]^{-1} B X_s \tag{3.2}$$

provided that the inverse of $\left[1-A\right]$ exists. Thus, any dependent node variables X_j may be expressed in terms of the source node variables in the form

$$X_j = T_{j1} X_{s1} + T_{j2} X_{s2} + \ldots\ldots + T_{jm} X_{sm} \tag{3.3}$$

where T_{ji} is called the transmission from the source node X_{si} to the dependent node X_j. T_{ji} can be obtained by the following formula

$$T_{ji} = \left.\frac{X_i}{X_{si}}\right|_{X_{si}\neq 0;\ X_{s1}=0,\ X_{s2}=0,\ \ldots\ldots,\ X_{sm}=0}$$

$$= \frac{1}{\Delta}\sum_{k} F_k \Delta_k \tag{3.4}$$

where $\Delta = \sum_{N=0}^{\infty} (-1)^N L(N)$ (3.5)

and define zero order loop as

$L(0) = 1$ for mathematical convenience,

$L(1)$ = sum of all the first order loop weights,

$L(N)$ = sum of all N-th order loop weights.

F_k = weight of the k-th path from the source node X_{si} to dependent node X_j.

Δ_k = sum of those terms in Δ without any constitute loops touching F_k.

where path and loop are defined as follows:

Path - A path from any node X_i to node X_j is any route leaving X_i and terminating at node X_j along which no node is encountered more than once.

Loop - A path whose initial node and terminal node coinicide.

Path weight - The product of all branch weights in a path.

Loop weight - The product of all branch weights in a loop.

n-th order loop - A set of n disjointed loops.

The summation is taken over all paths from X_{si} to X_j. The preceding rule is called Mason's rule, and Eq.(3.4) is named Mason's formula [11].

Now we apply the closed system idea to cope with the evaluation of the denominator and numerator by a single process. Suppose that we wish to find T_{ij} (X_j due to source-node variable X_{si}), we first add an additional branch with branch weight of -P and directed from node X_j to node X_{si}. Then we evaluate the system determinant Δ_{cs} according to Eq.(3.5) and sort the result into two groups under the condition that whether each term contains the symbol P. Thus Mason's formula becomes

$$\Delta_{cs} = \Delta + P \sum_{k} F_k \Delta_k \qquad (3.6)$$

To change a network to a signal flow graph, we first select a tree from the given network such that all voltage sources are tree branches, and all current sources are cotree branches. In order to reduce the number of nodes in SFG, we usually form the compact signal flow graph by the following procedures:

1. For each immittance cotree branch Y_k, express its current in terms of tree branch voltages by the use of $I_k = Y_k V_k$ and the Kirchhoff's voltage law equation.
2. For each immittance tree branch Z_j, express its voltage in terms of cotree branch currents by use of $V_j = Z_j I_j$ and the Kirchhoff's current law equation.

Example 3.1

Consider the simple network shown in Fig. 3.1.

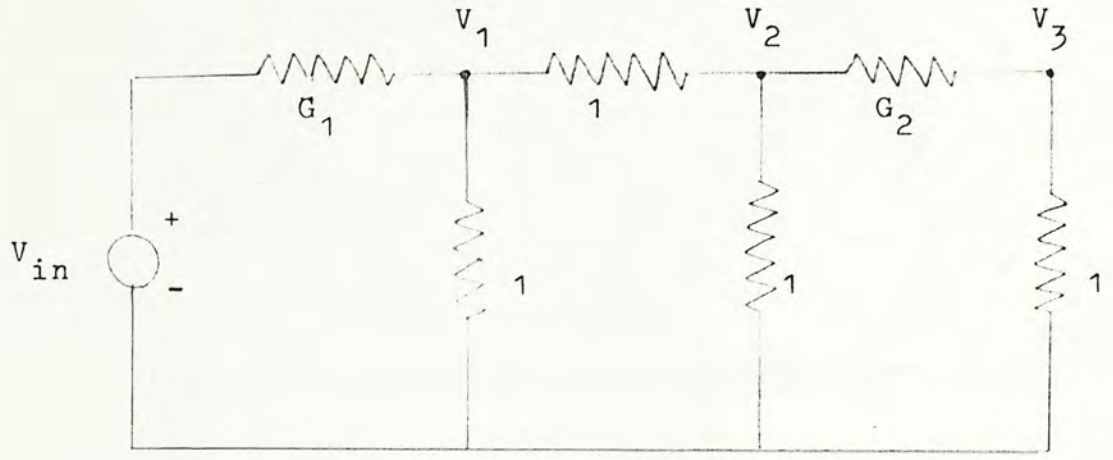


Fig. 3.1 : A simple network.

Applying the two procedures for formulating the signal flow graph, we can obtain the following set of equations.

$$
\begin{aligned}
I_1 &= (V_{in} - V_1)\,G_1, & V_1 &= I_1 - I_2 \\
I_2 &= V_1 - V_2, & V_2 &= I_2 - I_3 \\
I_3 &= (V_2 - V_3)\,G_2, & V_3 &= I_3
\end{aligned}
\qquad (3.7)
$$

From Eq.(3.7), we can obtain the signal flow graph representing the network. Now, we apply the closed system idea by adding a branch with branch weight -P directed from the output V_3 to input V_{in}, the signal flow graph is shown in Fig. 3.2.

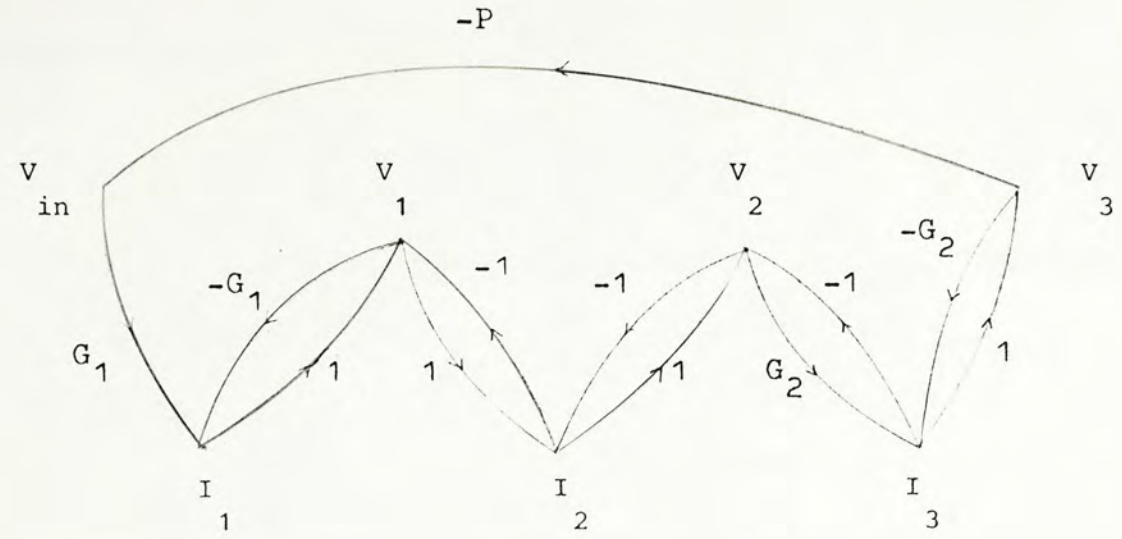


Fig. 3.2 : Signal flow graph represents the network shown in Fig. 3.1.

From the signal flow graph, we can generate all the first order loops as follows:

$$L_1 = (G_1)(1)(1)(1)(G_2)(1)(-P) = -P G_1 G_2$$

$$L_2 = (1)(-G_1) = -G_1$$

$$L_3 = (1)(-1) = -1$$

$$L_4 = (1)(-1) = -1$$

$$L_5 = (-1)(G_2) = -G_2$$

$$L_6 = (-G_2)(1) = -G_2 \qquad (3.8)$$

By applying Eq.(3.5) to evaluate the closed system determinant, we have

$$\Delta_{cs} = 1 - (L_1 + L_2 + L_3 + L_4 + L_5 + L_6) + (L_2 L_4 + L_2 L_5 + L_2 L_6 + L_3 L_5 + L_3 L_6 + L_4 L_6) - (L_2 L_4 L_6) \qquad (3.9)$$

Substituting Eq.(3.8) into Eq.(3.9), we have

$$\Delta_{ca} = 3 + 2G_1 + 5G_2 + 3G_1G_2 + PG_1G_2 \qquad (3.10)$$

Seperating the expression into numerator and denominator, we obtain the following network function:

$$\frac{V_3}{V_{in}} = \frac{G_1G_2}{3 + 2G_1 + 5G_2 + 3G_1G_2} \qquad (3.11)$$

The modification to SFG is slight but the improvement is significant.

Parameter Extraction Method

The first step to apply parameter extraction method [5] is the generation of indefinite admittance matrix (IAM) from a network, IAM of any n-terminal network (without independent sources inside the network), denoted by Y is an nxn matrix relating the terminal currents I and terminal voltages V as

$$I = Y V \qquad (3.12)$$

where the voltages V are measured with respect to a point external to the network.

The IAM has some special properties,

1. The sum of the elements in each row or column is zero. Hence the determinant of Y is zero.
2. All cofactors of Y are equal.
3. The node admittance matrix may be obtained from Y by deleting row k and column k, if node k of N is chosen to be the reference node.
4. Any circuit element will appear in exactly four positions in IAM.

By using the closed system technique described in the previous chapter, the network function can be obtained by sorting the expansion of cofactor of Y with respect to the existance of additional symbol which changes independent source to dependent source.

The cofactor of Y can be expanded by applying the extraction process to extract the symbol X in Y according to the formula as follow:

$$\mathrm{cof}(Y) = \mathrm{cof}(Y\big|_{X=0}) + (-1)^{j+m} X\, \mathrm{cof}(Y_X) \tag{3.13}$$

where X is the symbol appearing in Y at the position of row i,j and column k,m. Y_X is an IAM after extracting X, and is obtained by modifying Y as follow:

1. Add j-th row to i-th row.
2. Add m-th column to k-th column.
3. Delete j-th row and m-th column.

If $Y\big|_{X=0}$ and Y_X in Eq.(3.13) still contain other symbols, then by repeatedly applying the parameter extraction procedure to extract all the symbols, finally we obtain an expression of cofactor Y as

$$\mathrm{cof}(Y) = \sum_j P_j\, \mathrm{cof}(Y_j) \tag{3.14}$$

where the summation sums all the possible symbolic terms P_j, each P_j is symbolic terms including the sign, $\mathrm{cof}(Y_j)$ is the coefficient of the symbolic term P_j and Y_j is indefinite admittance matrix containing only numerical entries.

Example 3.2

We use Parameter Extraction method to generate the network function of the same network in Example 3.1. In order to form the indefinite admittance matrix, the voltage source is converted into current source, and the modified network is shown in Fig. 3.3.

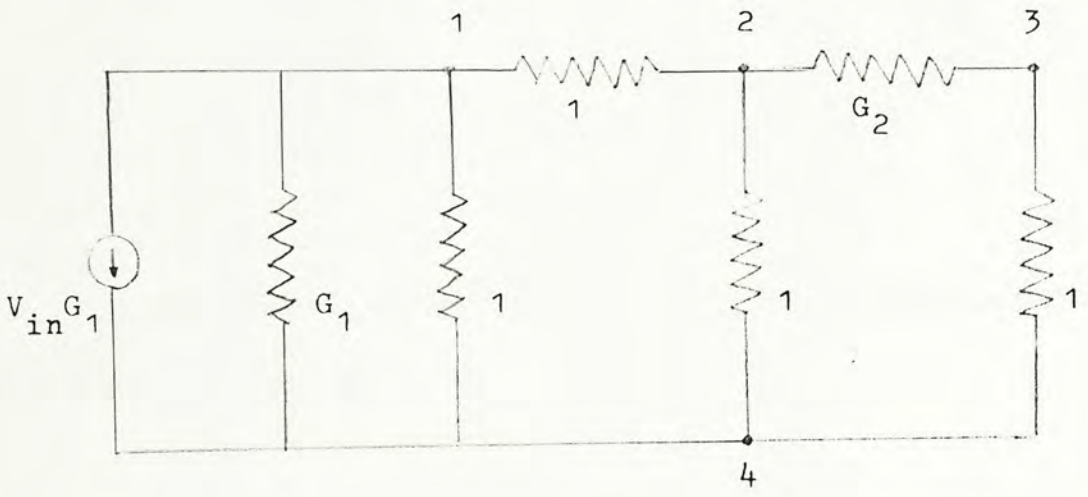


Fig. 3.3 : The modified network of Fig. 3.1.

Applying the closed system technique, we change the independent source to a dependent source related to the output V_{34}, such that

$$V_{in} G_1 = -P V_{34} \tag{3.15}$$

Therefore the indefinite admittance matrix is formed as follows:

$$Y = \begin{bmatrix} 2 + G_1 & -1 & P & -(1 + P + G_1) \\ -1 & 2 + G_2 & -G_2 & -1 \\ 0 & -G_2 & 1 + G_2 & -1 \\ -(1 + G_1) & -1 & -(1 + P) & 3 + P + G_1 \end{bmatrix} \tag{3.16}$$

Applying Eq.(3.13), we first extract the symbol P:

$$\mathrm{cof}(Y) = \mathrm{cof}\begin{bmatrix} 2+G_1 & -1 & 0 & -(1+G_1) \\ -1 & 2+G_2 & -G_2 & -1 \\ 0 & -G_2 & 1+G_2 & -1 \\ -(1+G_1) & -1 & -1 & 3+G_1 \end{bmatrix} + P\ \mathrm{cof}\begin{bmatrix} 1 & -2 & 1 \\ -1 & 2+G_2 & -(1+G_2) \\ 0 & -G_2 & G_2 \end{bmatrix} \tag{3.17}$$

Since symbols still exist in the cofactors, we apply Eq.(3.13) again to extract the symbol G_1 :

$$\mathrm{cof}(Y) = \mathrm{cof}\begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2+G_2 & -G_2 & -1 \\ 0 & -G_2 & 1+G_2 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix} + G_1\ \mathrm{cof}\begin{bmatrix} 3 & -2 & -1 \\ -2 & 2+G_2 & -G_2 \\ -1 & -G_2 & 1+G_2 \end{bmatrix}$$

$$+ P\ \mathrm{cof}\begin{bmatrix} 1 & -2 & 1 \\ -1 & 2+G_2 & -(1+G_2) \\ 0 & -G_2 & G_2 \end{bmatrix} \tag{3.18}$$

Applying Eq.(3.13) again to extract the last symbol G_2, we have

$$\text{cof}(Y) = \text{cof}\begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix} + G_2 \text{ cof}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 3 & -2 \\ -1 & -2 & 3 \end{bmatrix} + G_1 \text{ cof}\begin{bmatrix} 3 & -2 & -1 \\ -2 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

$$+ G_1 G_2 \text{ cof}\begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix} + P \text{ cof}\begin{bmatrix} 1 & -2 & 1 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} + G_2 P \text{ cof}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad (3.19)$$

Evaluating all the numerical cofactors gives the result

$$\text{cof}(Y) = 3 + 5 G_2 + 2 G_1 + 3 G_1 G_2 + P G_2 \qquad (3.20)$$

Seperating the expression into numerator and denominator, we obtain the following network function:

$$\frac{V_{34}}{I_{in}} = \frac{V_{34}}{G_1 V_{in}} = \frac{G_2}{3 + 2 G_1 + 5 G_2 + 3 G_1 G_2}. \qquad (3.21)$$

Chapter 4. Generation of Symbolic Network Functions via Symbolic Manipulation by Ordered Binary Tree Data Structure

In this chapter, we first review Gaussmann Algebra which is a powerful tool to expand a sparse determinant. The new method is then presented: it is a symbolic manipulation process to simplify a symbolic expression by hierarchical tree structure. Application of this symbolic manipulation together with Gaussmann Algebra and Signal Flow Graph method to generate network function is finally discussed.

Evaluation of Determinant by Gaussmann Algebra [13]

The basic principle of Gaussmnan Algebra is to find out all the terms which are the product of non-zero entries at different row and different column in the determinant. We first keep the column numbers in natural order and then find all the possible combination of row numbers. Suppose we want to evaluate the following determinant:

$$X = \begin{vmatrix} x_{11} & x_{12} & \cdots & x_{1,n-1} & x_{1,n} \\ x_{21} & x_{22} & \cdots & x_{2,n-1} & x_{2,n} \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{n,n-1} & x_{n,n} \end{vmatrix} \qquad (4.1)$$

Expressing the columns in vector notation gives

$$X = \begin{vmatrix} \underline{x}_1 & \underline{x}_2 & \cdots & \underline{x}_{n-1} & \underline{x}_n \end{vmatrix} \qquad (4.2)$$

where $\underline{x}_i$ is the column vector of X at the i-th column.

Then the determinant X is defined as the exterior product of all the column vectors,

$$X = \underline{x}_1 \wedge \underline{x}_2 \wedge \cdots \wedge \underline{x}_{n-1} \wedge \underline{x}_n \tag{4.3}$$

where $\wedge$ denoted the exterior product under the following rules:

$$\underline{x}_i \wedge \underline{x}_i = 0 \tag{4.4a}$$

$$\underline{x}_i \wedge \underline{x}_j \wedge \underline{x}_i = 0 \tag{4.4b}$$

$$(\underline{x}_i \wedge \underline{x}_j) \wedge \underline{x}_k = \underline{x}_i \wedge (\underline{x}_j \wedge \underline{x}_k) \tag{4.4c}$$

$$\underline{x}_i \wedge (\underline{x}_j + \underline{x}_k) = \underline{x}_i \wedge \underline{x}_j + \underline{x}_i \wedge \underline{x}_k \tag{4.4d}$$

Finally we can determine the sign of a term by simply checking the odd or even permutation of the number of interchanges of the row numbers into natural order. Even permutation implies positive term while odd permutation implies negative term.

Example 4.1

Consider the following determinant

$$A = \begin{vmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{vmatrix} \tag{4.5}$$

and there may be numerical values or symbolic parameters in each non-

zero entries. Denote $\underline{a}_i$ as the set of row numbers of all the non-zero entries of the i-th column vector, such that

$$\underline{a}_1 = (1 , 2)$$

$$\underline{a}_2 = (1 , 2 , 3)$$

$$\underline{a}_3 = (2 , 3) \qquad (4.6)$$

Applying Eq.(4.3), the determinant A becomes

$$A = \underline{a}_1 \wedge \underline{a}_2 \wedge \underline{a}_3 \qquad (4.7)$$

Substituting Eq.(4.6) into Eq.(4.7) and applying Eq.(4.4), we have

$$\begin{aligned} A &= (1,2) \wedge (1,2,3) \wedge (2,3) \\ &= (11,12,13,21,22,23) \wedge (2,3) \\ &= (12,13,21,23) \wedge (2,3) \\ &= (122,123,132,133,212,213,232,233) \\ &= (123,132,213) \qquad (4.8) \end{aligned}$$

The terms 132 and 213 are negative since there is one interchange of positions of digits to arrange the numbers into natural order.

In terms of entries of the determinant, we have

$$A = a_{11} a_{22} a_{33} - a_{11} a_{32} a_{23} - a_{21} a_{12} a_{33} \qquad (4.9)$$

Let $a_{11} = 1 + R_1$, $a_{12} = -R_1$

$a_{21} = -R_1$, $a_{22} = 1 + R_1 + R_2$, $a_{23} = -R_2$

$a_{32} = -R_2$, $a_{33} = 2 + R_2$ (4.10)

Substituting Eq.(4.10) into Eq.(4.9), we have

$$A = (1+R_1)(1+R_1+R_2)(2+R_2) - (1+R_1)(-R_2)(-R_2) - (-R_1)(-R_1)(2+R_2) \tag{4.11}$$

Eq.(4.11) is the result to expand the symbolic determinant, A, by applying Gaussmann Algebra only. The expression of the result is still very complicated without simplification. After simplifying Eq.(4.11), the final expression of the determinant becomes

$$A = 2 + 4R_1 + 3R_2 + 4R_1R_2 \tag{4.12}$$

Simplification of the symbolic expression is very important in generation of symbolic network functions, because the unsimplified result is still very complicated and we cannot obtain the advantages described in [1] for the symbolic network functions without simplification. However the procedure to simplify Eq.(4.7) to Eq.(4.8) is not a simple task to be handled with a digital computer. We need an efficient symbolic manipulation method to handle the problem of cancellation of repeated symbolic terms.

The New Symbolic Manipulation Method by Ordered Binary Tree Data Structure

In symbolic network analysis, large amount of effort is used to cancel the repeated symbolic terms. With the aid of an efficient symbolic manipulation method, we may generate symbolic network function easily by direct expansion of a symbolic determinant. The most widely used symbolic manipulation methods are integer code methods [2] [14] which are applied in signal flow graph to cancel the repeated symbolic

terms. The integer code methods transform the symbols to some equivalent integer values so that multiplication of symbols becomes operation of integer values, therefore the integer code methods provide the representation of product of different symbols.

In general, a symbolic expression is represented as sum of products of symbols. To represent a symbolic expression in another form, we need the representation of both addition and multiplication of symbolic terms. The integer code methods only transform the multiplication of symbols to operation of integer values. Another method is needed to represent the summation between symbolic terms. The array data structure is the simplest method, but searching, insertion and deletion of terms are difficult. The link list data structure is much more flexible, but it has a serious drawback: when a lot of symbolic terms appear in the link list, the searching time of a term will be large.

In order to reduce the searching time, an ordered binary tree data structure is used. An ordered tree is a tree in which the relative order of subtree is important. The details of link list and tree data structure can be found in any reference about data structure such as [15] [16]. The idea of using ordered binary tree for representing symbolic expression is new.

Each node of the binary tree consisting four subfields represents a symbolic term. The four subfields are shown as follows:

1. VL : value of the coefficient of the symbolic term,
2. NO : symbol number of the symbol represented by this node,
3. VT : vertical branch points to the node of next higher level, and
4. HT : horizontal branch points to the node of same level.

While each node represents one symbolic term, the whole symbolic expression is formed by summing up all the symbolic terms represented by the nodes. In order to represent different combination of symbolic terms, the two branches, VT and HT have different meanings.

Meaning of Vertical Branch

The vertical branch VT, contains the meaning of multiplication of symbols. If a node is connected by VT, then the symbolic term represented by this node is the product of symbol of this node and the symbolic term represented by the predecessor, or its father. For example, the tree structure representing the symbolic expression

$$3 + X_1 + 2X_1^2 + 4X_1^2X_2 + X_1^2X_2X_3$$

is shown in Fig. 4.2.

In Fig. 4.2, node A is the root of the tree and it only represents the constant term. Node B represents the symbolic term X_1 with coefficient 1. The symbol of node C is also X_1, and this node is connected by vertical branch VT of node B. Therefore node C represents the product of the symbol of node C and the symbolic term represented by node B. Together with the coefficient, the node represents the term $2X_1^2$. The symbolic term represented by node D is $X_1^2X_2$ which is formed by multiplying the symbol of node D, X_2 to the symbolic term represented by node C, X_1^2. Together with the coefficient, node D represents the

symbolic term $4 X_1^2 X_2$. Similarly node E represents the term $X_1 X_2^2 X_3$.

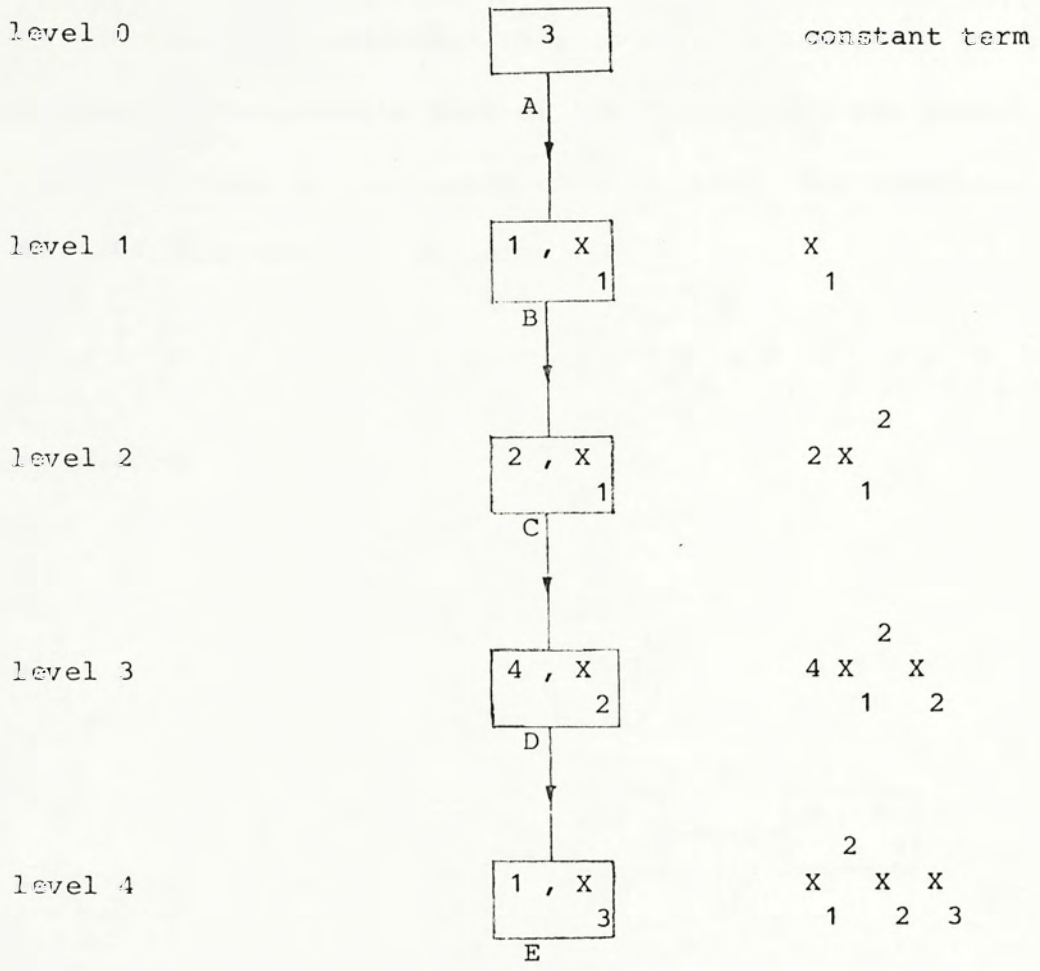


Fig. 4.2 : Tree structure representation of the symbolic expression

$$3 + X_1 + 2 X_1^2 + 4 X_1 X_2^2 + X_1 X_2 X_3^2 .$$

Meaning of Horizontal Branch

By using the vertical branches, we can only represent products of some symbols. In order to represent different combinations of symbolic terms, the horizontal branch HT which contains the meaning of replacement of a symbol, is needed. If a node is connected by HT, then the node represents the symbolic term of its father with the symbol of its father being replaced by the symbol of this node. For example, the representation of the symbolic expression

$$3 + X_1 + 4X_2 + 2X_3 + X_1^2 + 4X_1X_2 + X_1X_2 + X_1^2X_2 + X_1X_2^2 + X_1X_2X_3$$

is shown in Fig. 4.3.

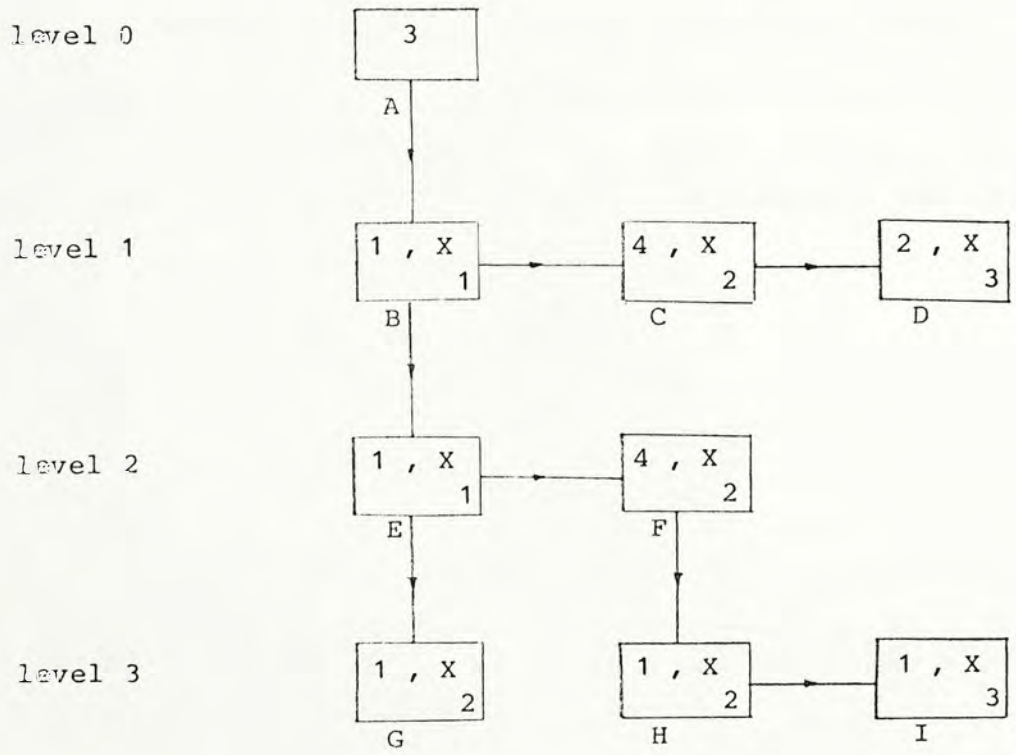


Fig. 4.3 : Tree structure representation of the symbolic expression

$$3 + X_1 + 4X_2 + 2X_3 + X_1^2 + 4X_1X_2 + X_1X_2 + X_1^2X_2 + X_1X_2^2 + X_1X_2X_3 .$$

In Fig. 4.3, node C is connected by HT of node B, therefore the symbol of node B, X_1 is replaced by the symbol of node C, X_2. Together with the coefficient, node C represents the term $4X_2$. Also node D is connected by HT of node C, so that X_2 is replaced by X_3, Together with the coefficient, node D represents the term $2X_3$. Node F is connected by HT of node E which represents the symbolic term X_1^2, but only the symbol of node E is replaced by symbol of node F, also the meaning of multiplication for the vertical branch between node B and node E is still hold. Therefore together with the coefficient, node F represents the symbolic term $4X_1X_2$. Finally, the symbol of node H, X_2 is replaced by the symbol of node I, X_3, and node I represents the symbolic term $X_1X_2X_3$.

By using the above representation of symbolic expression, the nodes of the ordered binary tree are divided into different levels. (The meaning of level used here is different from the usual meaning of level in tree data structure.) The level number of a node is equal to the number of symbols in a symbolic term represented by this node. For example, in Fig. 4.3 node A is the root of the tree, which represents the constant term, therefore it is in level 0. Node B, C and D represent the symbolic terms X_1, X_2 and X_3 respectively, all of them are symbolic terms with only one symbol, therefore they are in level 1. Node E and F represent the symbolic terms X_1^2 and $X_1 X_2$, which are products of two symbols so that they are in level 2. Node G, H and I are in level 3 since they represent the symbolic terms $X_1^2 X_2$, $X_1 X_2^2$ and $X_1 X_2 X_3$ respectively and all of them consist of three symbols.

Therefore the nodes connected by HT are in the same level as their father, while the level of nodes connected by VT is in the next higher level of its father. And the level number is equal to the number of symbols in a symbolic term.

Insertion of a symbolic term

The procedure for insertion of a symbolic term to the ordered binary tree consists of two steps:

1. Sort the symbol numbers in ascending order.

In order to provide a unique representation for the symbolic expression, the symbols are assigned with symbol numbers and the

symbol numbers of a symbolic term are sorted in ascending order before inserted into the ordered binary tree. For example, the symbolic term $X_1^2 X_2$ are sorted as $X_1 X_1 X_2$ instead of $X_1 X_2 X_1$ or $X_2 X_1 X_1$ before inserted into the result tree.

2. Search the existance of the symbolic term in the binary tree.

Match all the symbols of the sorted symbolic term in the corresponding levels of the binary tree. If the symbolic term already existed, then we can simply insert the symbolic term by adding the corresponding coefficients, else we must add a new node in order to insert the new symbolic term.

For example, the result tree after adding the symbolic terms $3 X_1 X_2$, $3 X_1^3$ and X_3^3 to the symbolic expression represented in Fig. 4.3 is shown in Fig. 4.4. And the result becomes

$$3 + X_1 + 4 X_2 + 2 X_3 + X_1^2 + 7 X_1 X_2 + 3 X_1^3 + X_1^2 X_2 + X_1 X_2^2 + X_1 X_2 X_3 + X_3^3 .$$

In Fig. 4.4, since the symbolic term $X_1 X_2$ is already existed, it is added by changing the coefficient of node F. A new node P is used to represent the symbolic term $3 X_1^3$, it replaces the position of symbolic term $X_1^2 X_2$ which is now connected by HT of node P. Two new nodes R and

S are added in order to represent the symbolic term X_3^3. Node R is a dummy node with zero coefficient representing the symbolic term X_3^2 in order to produce the term X_3^3.

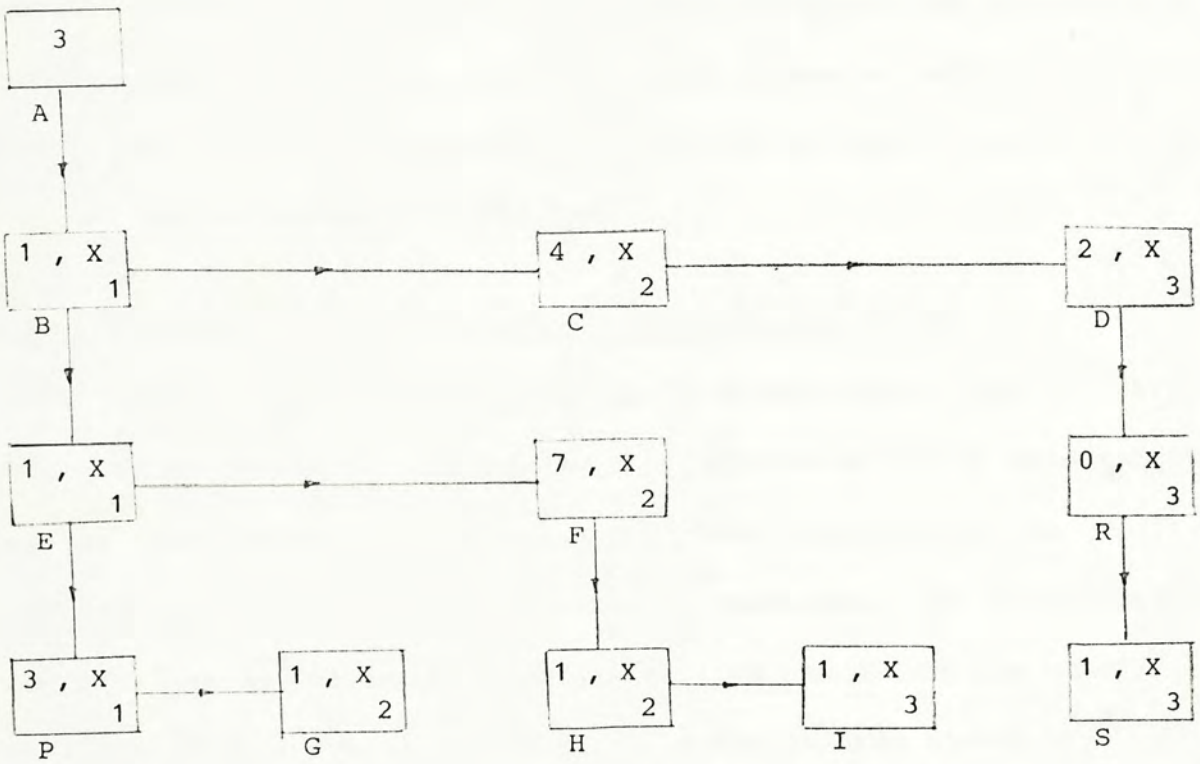


Fig.4.4 : The result tree after adding the symbolic terms $3 X_1^3 X_2$, $3 X_1^3$ and X_3^3 to the expression of Fig. 4.3. The result expression is

$$3 + X_1 + 4 X_2 + 2 X_3 + X_1^3 + 7 X_1 X_2 + 3 X_1^3 + X_1^2 X_2 + X_1 + X_2^2 + X_1 X_2 X_3 + X_3^3 .$$

Searching time of a symbolic term in a tree

A symbolic expression is uniquely represented by a hierarchical data structure. Since the symbols are sorted before inserted into the result tree, the symbol number of a node will never be smaller than symbol number of its father. Also the symbolic terms with n symbolic elements are put to the level n; the searching time of a particular term in the tree is much less than that required for the case of link list. The maximum number of nodes to be searched to determine the existance of a symbolic term is only n+r, where n is the number of symbols in the symbolic term (ie. the level number), and r is the symbol number of the last symbol in the sorted symbolic term.

Inverse Transform of tree structure to symbolic expression

The output stage to obtain the symbolic expression from the tree structure is extremely easy. The symbolic expression can be obtained by traversing the whole tree once and the tree traverse can be easily implemented by using recursive programming technique. By substituting the numerical values of symbolic parameters, we can obtain the numerical value of the network function directly from the ordered binary tree. The numerical value of the symbolic term represented by each node can be obtained by finding the numerical product of the symbol of the current node and the symbolic term represented by its father.

Arithematic Operations of Symbolic Expression in terms of Binary Tree Data Structure

The symbolic expression representation by ordered binary tree data structure is a general symbolic manipulation method, which is best applied in the field of arithematic of symbolic expressions.

1. Addition or Subtraction : Add(TREE1,TREE2) or Sub(TREE1,TREE2)

The operation is to add TREE1 to TREE2 or subtract TREE1 from TREE2, and the final result is given in TREE2. We first search each node of TREE1 in TREE2. If a symbolic term is already existed, then the symbolic term is added or subtracted by performing the corresponding operation on the coefficients. Otherwise we may add or subtract the symbolic terms by copying the whole subtree. Since the symbolic terms are sorted before entered into the ordered tree, search of each node of TREE1 can be started from the last searched node, and the operation can be completed by traversing the two trees only once.

2. Multiplication : Mult(TREE1,TREE2,TREE3)

The operation is to multiply TREE1 and TREE2 together and the final result is give in TREE3. We multiple every node of TREE2 to the whole tree of TREE1. Therefore we need traverse TREE2 once and traverse TREE1 n times, where n is the number of non-zero nodes in TREE2.

The algorithms which implement the arithematic operations (addition, subtraction and multiplication) in terms of tree structure, are given in the appendices.

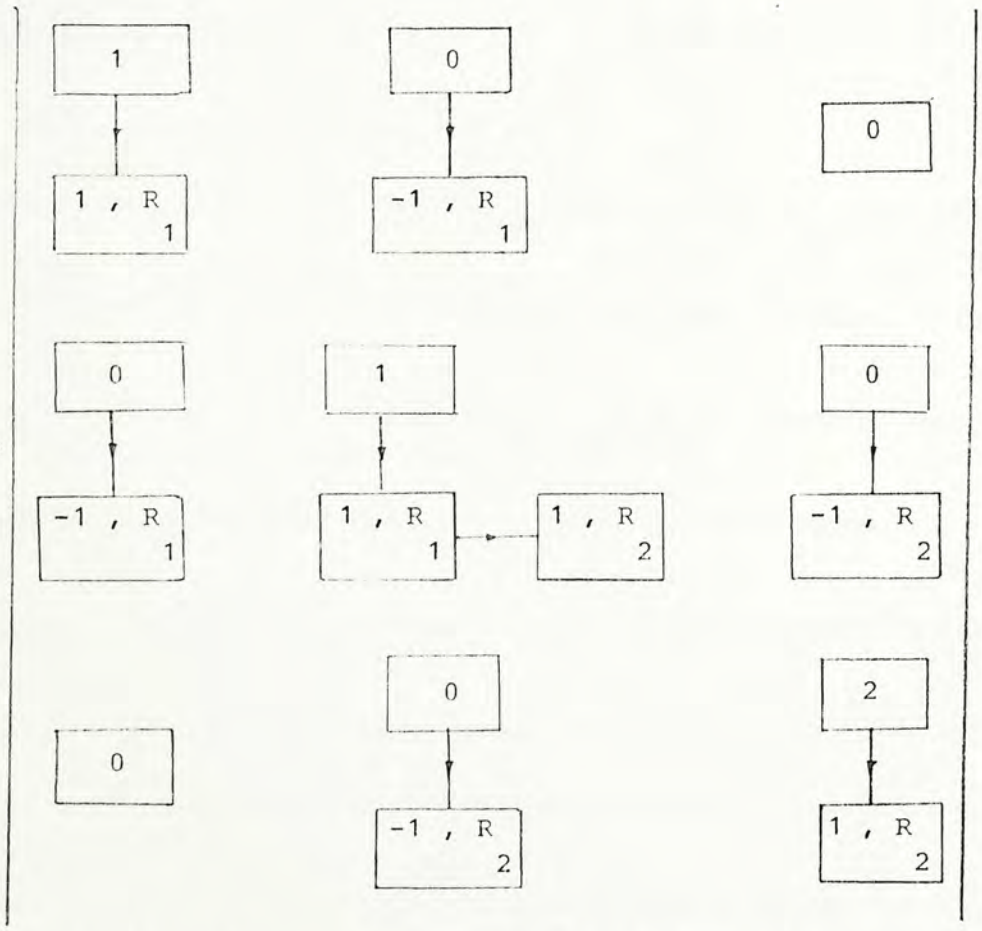
Application of tree structure symbolic manipulation method together with Gaussmann Algebra to evaluate a symbolic determinant

In the following example, the tree structure symbolic manipulation method is applied together with the Gaussmann Algebra to evaluate the symbolic determinant given in Eq.(4.10). During the manipulation process, we can see how the symbolic manipulation method cancels the repeated symbolic terms in the evaluation of the symbolic determinant.

Example 4.2

In order to evaluate the determinant, we first use tree structure to represent all the non-zero entries. The determinant is then written as follows:

A =

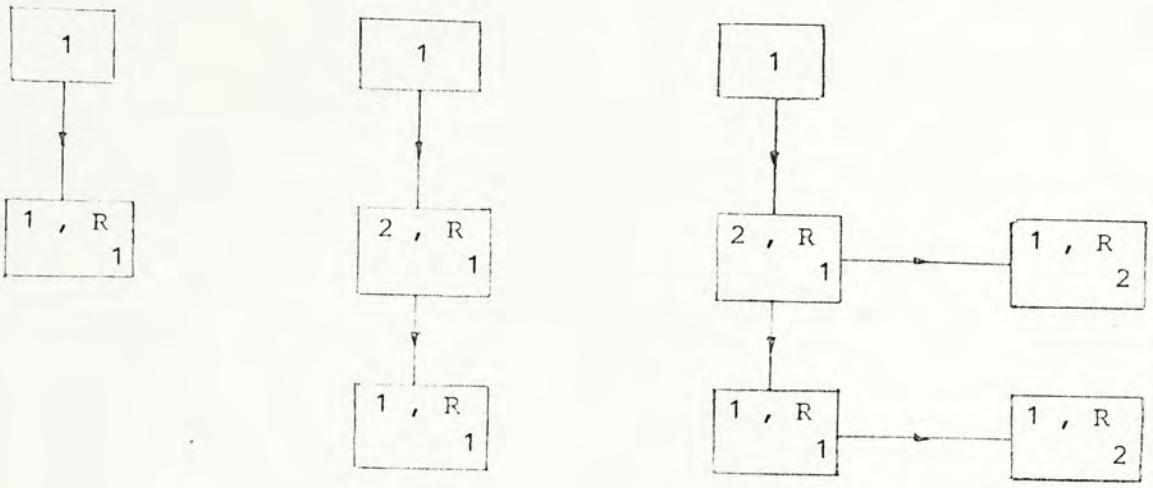


By applying Gaussmann Algebra, we evaluate A according to Eq.(4.9),

where $A = a_{11} a_{22} a_{33} - a_{11} a_{32} a_{23} - a_{21} a_{12} a_{33}$

We first generate $a_{11}\ a_{22}$, and the steps are shown as follows:

$a_{11}\ a_{22}$

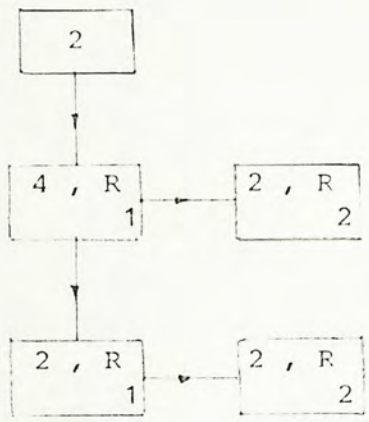


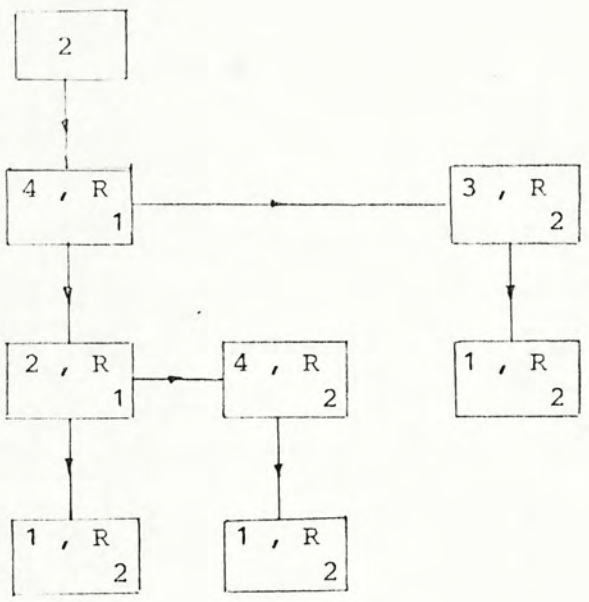
$1 + R_1$ $\qquad$ $1 + 2R_1 + R_1^2$ $\qquad$ $1 + 2R_1 + R_2 + R_1^2 + R_1 R_2$

The temporary result trees after each node of a_{22} multiplied to the whole tree of a_{11} are given above. When the constant term of a_{22} is multiplied to a_{11}, the first tree, $1 + R_1$ is formed. When the second node of a_{22} is multiplied to a_{11}, two symbolic terms, R_1^2 and R_1 are generated. Since R_1 is already existed in the temporary result tree, we can add this symbolic term by simply adding the coefficients, and we need to add a new node to the temporary result tree to represent the symbolic term R_1^2. The last tree shown above is the result after the third node of a_{22} is multiplied to a_{11}. Two more new nodes are used to

represent the symbolic terms R_2 and $R_1 R_2$. Finally we multiply a_{33} to the temporary result tree to obtain the product term $a_{11} a_{22} a_{33}$. The steps are shown as follows:

$a_{11} a_{22} a_{33}$





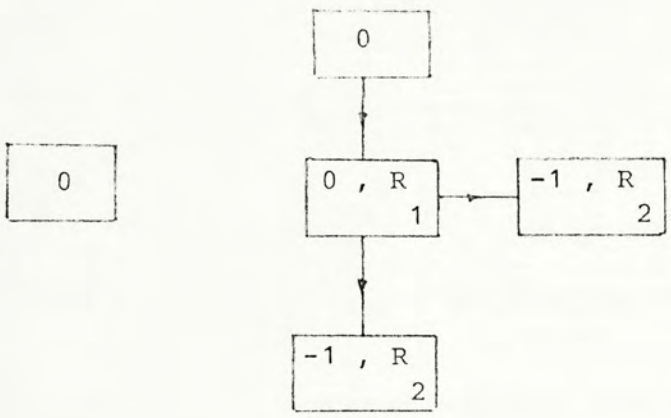
$$2 + 4R_1 + 2R_2 + 2R_1^2 + 2R_1 R_2$$

$$2 + 4R_1 + 3R_2 + 2R_1^2 + 4R_1 R_2 + R_2^2 + R_1 R_2^2 + R_1^2 R_2$$

The first tree shown above is the temporary result after the first node of a_{33} is multiplied to $a_{11} a_{22}$. The second one is the tree structure representing the product term $a_{11} a_{22} a_{33}$.

After generating the first product term, we need to evaluate the other two product terms according to Eq.(4.9). The second product term is $a_{11}\ a_{32}\ a_{23}$ and the steps are shown as follows:

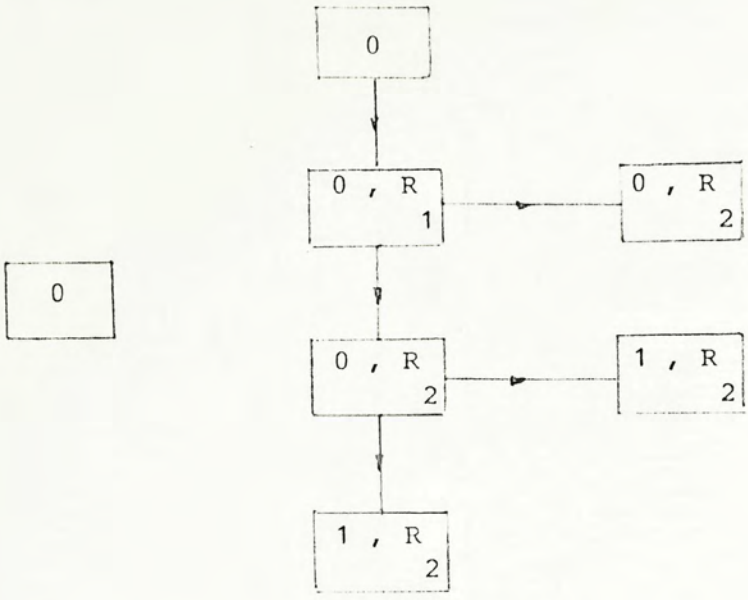
$a_{11}\ a_{32}$



$-R_2\ -R_1R_2$

The first temporary result tree with only one node of zero value is generated when the zero value of the constant term of a_{32} is multiplied to a_{11}. When the second node of a_{32} is multiplied to a_{11}, the second tree shown above is generated. Then we multiply a_{23} to this temporary result to obtain the product term $a_{11}\ a_{32}\ a_{23}$.

a_{11} a_{32} a_{23}



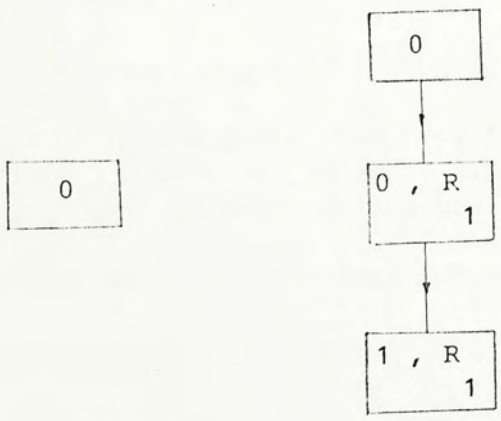
$R_2^2 + R_2 R_1 R_2$

The first tree is zero again because the constant term of a_{23} is zero.

In the second tree, dummy nodes with zero coefficient are generated in order to produce the symbolic terms R_2^2 and $R_2 R_1 R_2$.

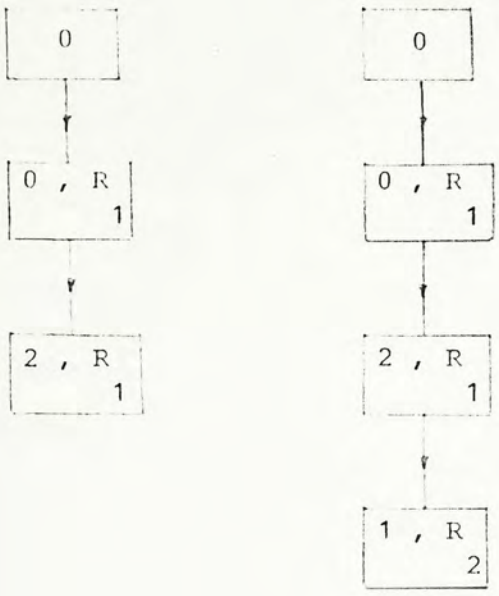
We find the last product term in the same way as follows:

a_{21} a_{12}



R_1^2

a_{21} a_{12} a_{33}



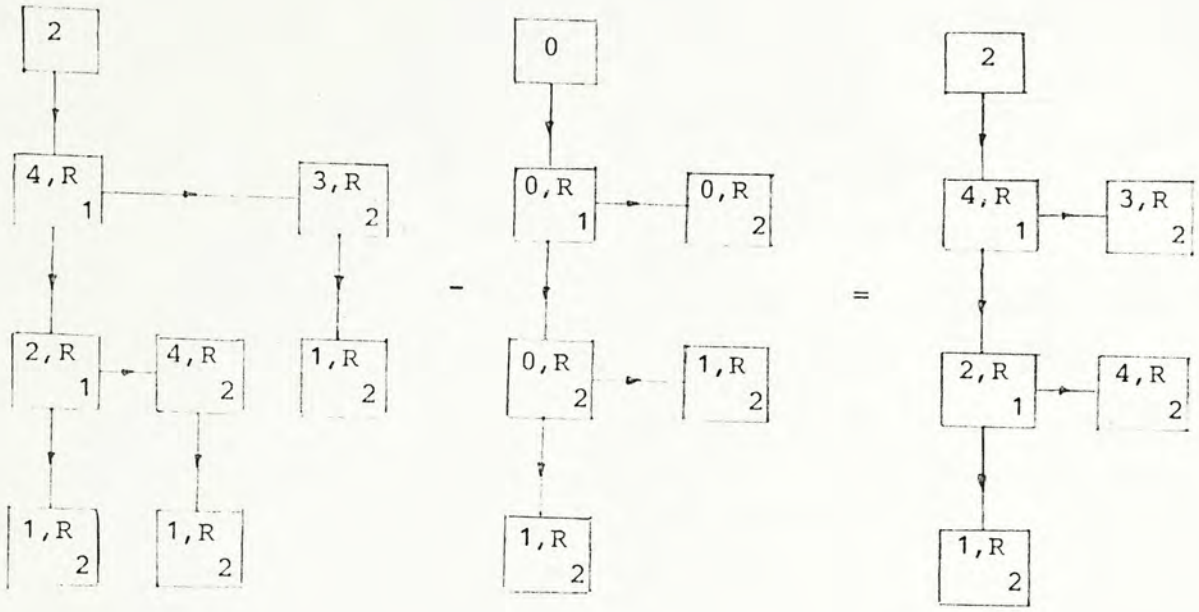
$2R_1^2$ $2R_1^2 + R_1 R_2^2$

After all the product terms have been found, we perform the subtraction between the product terms.

To perform the subtraction between two trees, we only need to change the coefficients of the first tree for the non-zero symbolic terms of the second one. The repeated symbolic terms will be cancelled after the subtraction. If the coefficient of a node is zero after the subtraction and the node does not consist of vertical branch, then the node can be deleted.

We first find the difference between the first two product terms, and the steps are shown as follows:

$a_{11}\ a_{22}\ a_{33} - a_{11}\ a_{32}\ a_{23}$

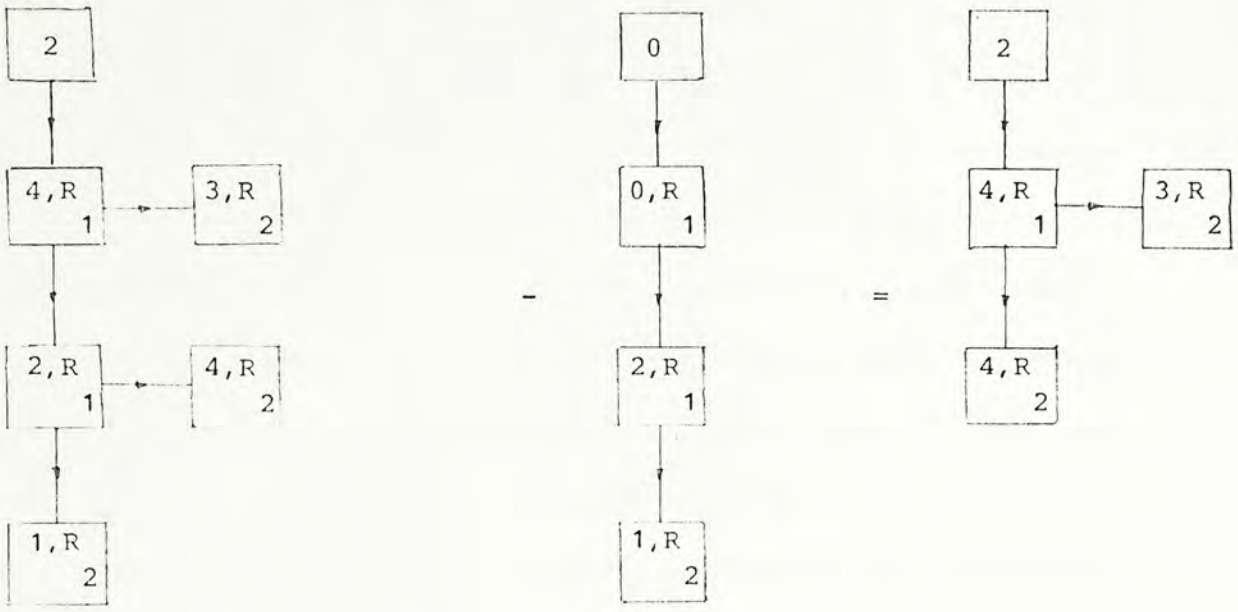


$$(2+4R_1+3R_2+2R_1^2+4R_1R_2+R_2^2+R_2^2+R_1R_2^2) - (R_2^2+R_1R_2^2) = 2+4R_1+3R_2+2R_1^2+4R_1R_2+R_1^2R_2^2$$

The subtaction is performed by changing the corresponding coefficients of the two non-zero symbolic terms R_2^2 and $R_1R_2^2$ in the first tree. The result coefficients are zero and these two nodes do not consist of vertical branch, therefore we can delete these two nodes to obtain a simpler tree.

According to Eq.(4.9), we need to perform one more subtraction to obtain the final result. The third product term is subtracted from the temporary result and the final result is shown in the following:

$$(a_{11}\,a_{22}\,a_{33} - a_{11}\,a_{32}\,a_{23}) - a_{12}\,a_{21}\,a_{33}$$



$$2+4R_1+3R_2+2R_1^2+4R_1R_2+R_1^2R_2 \;-\; 2R_1^2+R_1R_2^2 \;=\; 2+4R_1+3R_2+4R_1^2R_2$$

With the aid of the tree structure symbolic manipulation method, we can obtain the expansion of A as $2 + 4R_1 + 3R_2 + 4R_1^2R_2$ instead of the complicated result first obtained in Eq.(4.11).

Generation of Partially Symbolic Network Functions

In the above example, all the non-zero entries contain symbolic parameters, therefore all those entries must be represented by ordered binary trees. However we frequently encounter a large circuit with only a few circuit elements represented by symbols. For such a problem, a determinant with large dimension is correspondingly formed and there would be only a few entries containing symbolic parameters. For saving computer storage, it is advantageous to use two kinds of data types to represent the entries of the determinant. We use real number data type

to represent numerical values while use binary tree data structure to represent symbolic terms. The multiplication between numerical value and tree structure can be performed simply by multiplying the numerical value to the coefficient of every node of the tree structure. The operation can be done by traversing the tree once. In using the cofactor form expansion, we can extract all the symbolic entries by expanding the determinant at those entries. After all the symbolic entries are extracted, numerical determinants are left. Then we can use fast numerical algorithm to evaluate the numerical determinants. Finally we can obtain the expansion of symbolic determinant by multiplying the value of numerical determinant to the product of extracted entries.

Example 4.3

Consider the network shown in Fig. 4.5, only two circuit elements are represented in symbols, and all the other circuits elements are 1 ohm resistors.

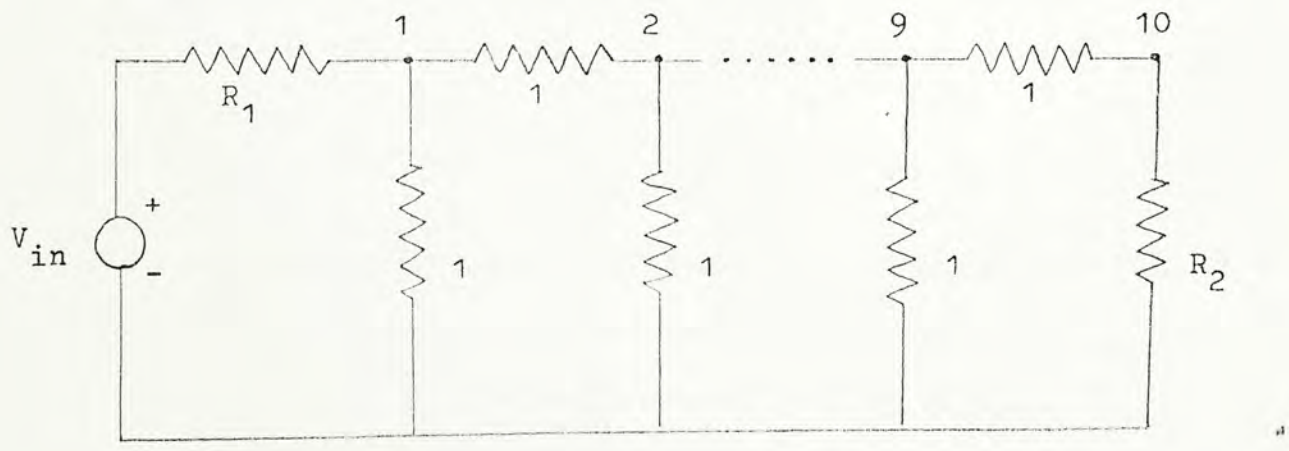


Fig. 4.5 : A network with 10 nodes, only R_1 and R_2 are symbols and all other circuit elements are 1 ohm resistors.

By applying loop analysis and the closed system technique, we can obtain the system equation as

$$Z \; I = 0 \tag{4.13}$$

where

$$Z = \begin{bmatrix} 1 + R_1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & P \\ -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 + R_2 \end{bmatrix} \tag{4.14}$$

In Z, there are only three entries consist of symbols, they are $z_{1,1}$, $z_{1,10}$ and $z_{10,10}$. To evaluate the system determinant Δ ($|Z|$), we only use tree data structure to represent these three entries, and use real data type to represent all the other entries. Then we can expand Δ by extracting the symbolic entries in cofactor form as follows:

$$\Delta = z_{1,1} \Delta_{1,1} - z_{1,2} \Delta_{1,2} - z_{1,10} \Delta_{1,10} \tag{4.15}$$

$$= (1 + R_1) \Delta_{1,1} - (-1) \Delta_{1,2} - P \Delta_{1,10} \tag{4.16}$$

Since symbolic entries still exist in $\Delta_{1,1}$ and $\Delta_{1,2}$, we apply the same procedure to extract $z_{10,10}$

$$\Delta = (1 + R_1)\left((1 + R_2)\left[\Delta_{1,1}\right]_{9,9} - \left[\Delta_{1,1}\right]_{9,8}\right) + (1 + R_2)\left[\Delta_{1,2}\right]_{9,9} - \left[\Delta_{1,2}\right]_{9,8} - P\Delta_{1,10} \qquad (4.17)$$

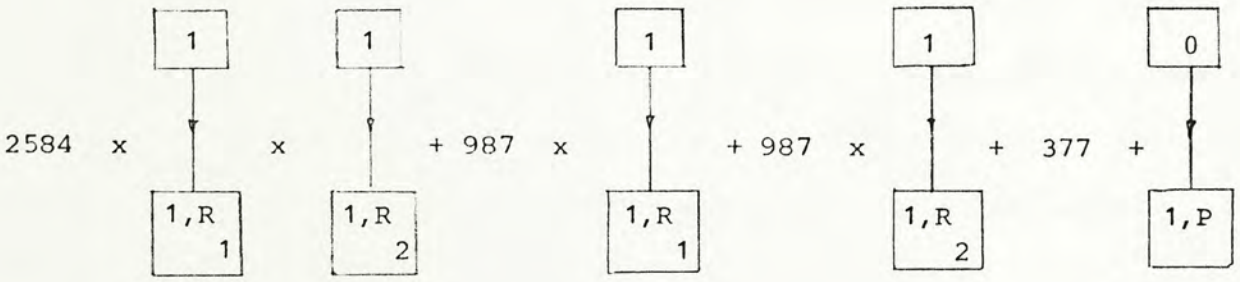
where $\left[\Delta_{i,j}\right]_{m,n}$ is the cofactor at m-th row and n-th column of $\Delta_{i,j}$.

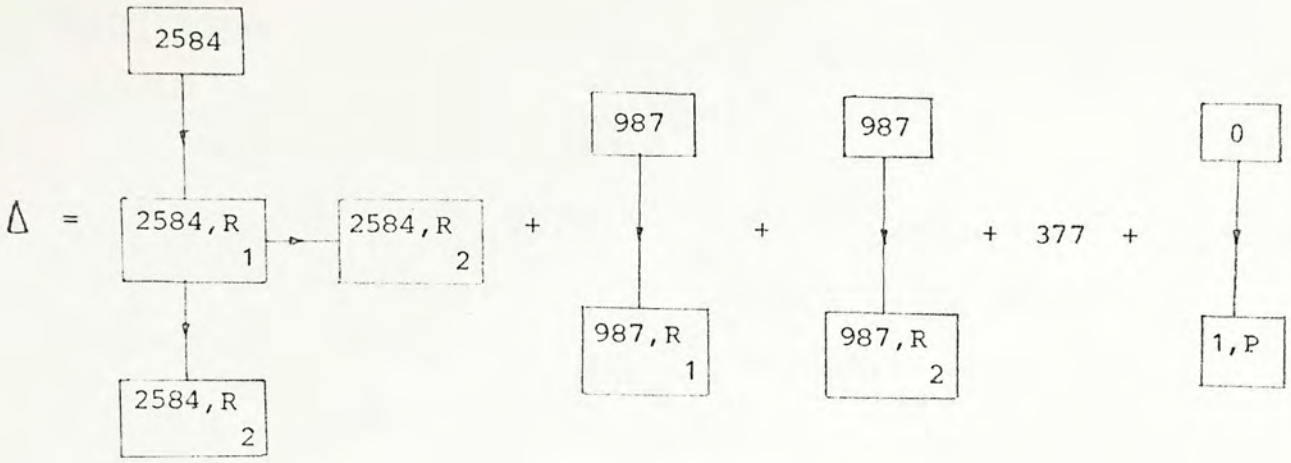
Since there is no more symbolic entry in all the cofactors in Eq.(4.17), we can evaluate the numerical determinants and obtain the following result:

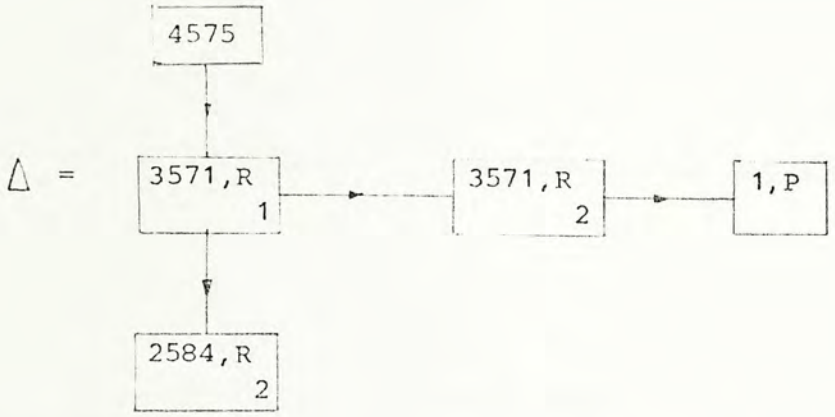
$$\Delta = 2584\ (1 + R_1)(1 + R_2) + 987\ (1 + R_1) + 987\ (1 + R_2) + 377 + P$$

Performing the operations on tree structrues gives

Δ =







Inverse transforming the tree structure to symbolic expression, we obtain the expansion of the determinant

$$\Delta = 4575 + 3571\, R_1 + 3571\, R_2 + 2584\, R_1 R_2 + P. \qquad (4.18)$$

In the above computation, there are only evaluation of five numerical determinants, one tree structure multiplication, three multiplications between numerical value and tree structure, and three tree structure additions. Therefore both computer time and storage can be saved.

Cancellation of repeated symbolic terms in Signal Flow Graph by tree structure symbolic manipulation method

It is known that in use of Signal Flow Graph, the problem of cancellation of repeated symbolic terms is a main drawback. In the following example, we can observe how the symbolic manipulation method helps in solving the cancellation problem in a Signal Flow Graph.

Example 4.4

Consider the circuit shown in Fig. 4.6, based on the circuit, we can obtain the set of equations in Eq.(4.19) and the signal flow graph in Fig. 4.7.

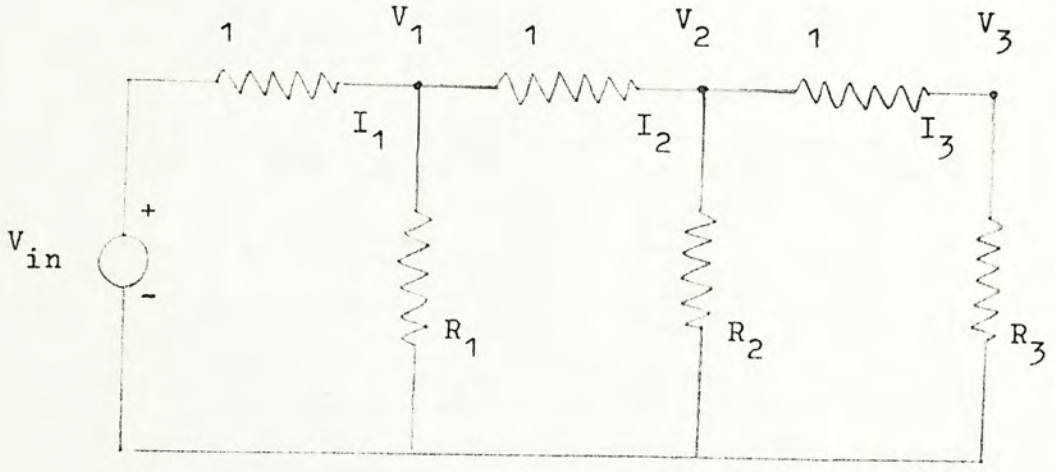


Fig. 4.6 : A simple circuit.

$$I_1 = V_{in} - V_1, \qquad V_1 = (I_1 - I_2) R_1$$

$$I_2 = V_1 - V_2 \qquad V_2 = (I_2 - I_3) R_2$$

$$I_3 = V_2 - V_3 \qquad V_3 = I_3 R_3 \qquad (4.19)$$

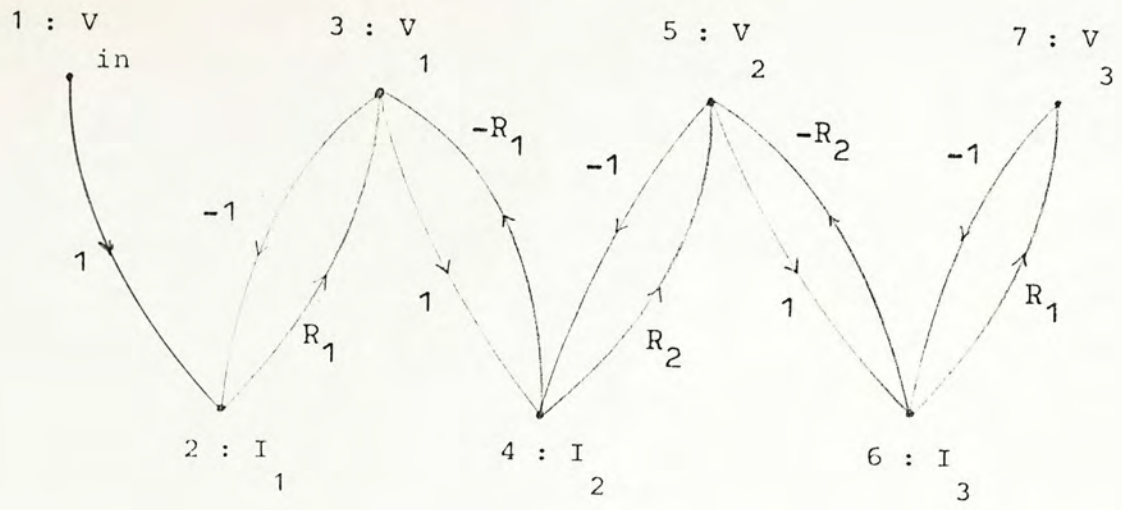


Fig. 4.7 : Signal flow graph obtained from the circuit in Fig. 4.5.

From the Fig. 4.7, we can obtain all the first order loops shown in Table 4.1.

Table 4.1 : First order loops.

loop number	node number in SFG	loop weights
1	2 3 2	$-R_1$
2	3 4 3	$-R_1$
3	4 5 4	$-R_2$
4	5 6 5	$-R_2$
5	6 7 6	$-R_1$

From the first order loops, we generate the second order loops and third order loops shown in Table 4.2 and Table 4.3.

Table 4.2 : Second order loops.

first order loop numbers	loop weights
1 3	$R_1 R_2$
1 4	$R_1 R_2$
1 5	R_1^2
2 4	$R_1 R_2$
2 5	R_1^2
3 5	$R_1 R_2$

Table 4 3 : Third order loops.

first order loop numbers	loop weights
1 3 5	$-R_1^2 R_2$

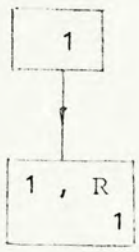
Applying Eq.(3.5), the system determinant becomes:

$$\Delta = 1 - L(1) + L(2) - L(3) \qquad (4.20)$$

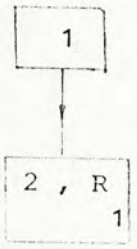
Without the aid of any symbolic manipulation method, the result will be

$$\Delta = 1 + (R_1 + R_1 + R_2 + R_2 + R_1) + (R_1 R_2 + R_1 R_2 + R_1^2 + R_1 R_1 + R_1^2 + R_1 R_2) + R_1^2 R_2 \qquad (4.21)$$

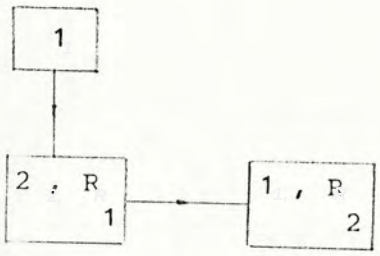
In order to simplify the result, we apply the tree structure symbolic manipulation to add all the loop weights. We first create a tree with only the root node of unit value, then we add all the first order loop weights. Five temporary result trees after adding each first order loop weight are shown as follows:



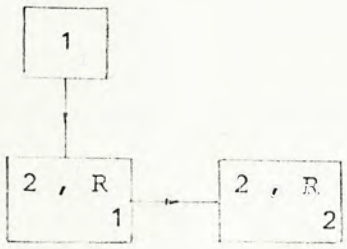
$1 + R_1$



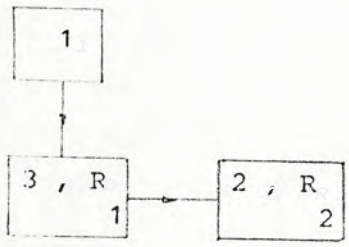
$1 + 2R_1$



$1 + 2R_1 + R_2$



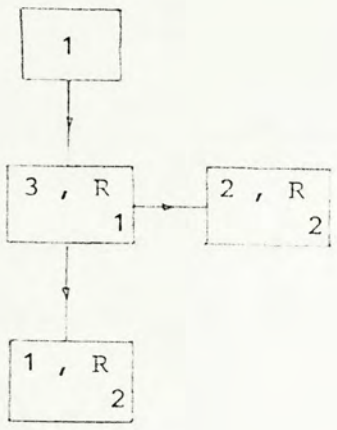
$1 + 2R_1 + 2R_2$



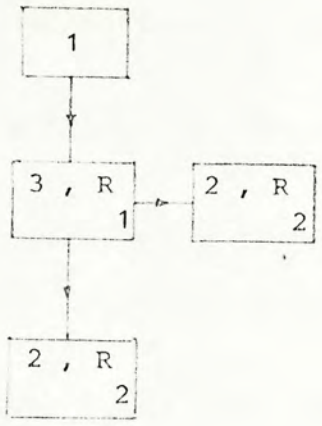
$1 + 3R_1 + 2R_2$

The first loop weight is R_1, a new node is created to represent this symbolic term. Since the second loop weight is also R_1, the term can be added by simply adding the coefficients. The third loop weight is R_2, a new node is created to represent this new symbolic term. The last two symbolic terms, R_1 and R_2 are then added by adding the corresponding

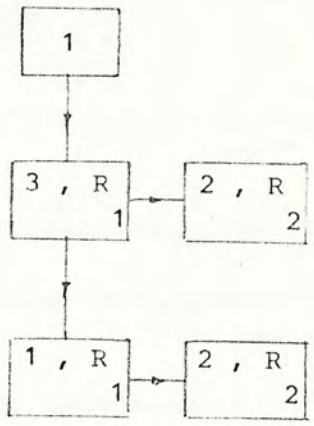
coefficient. After all the first order loop weights are added, we add the second order loop weights to the temporary result, the steps are shown in the following temporary result trees.



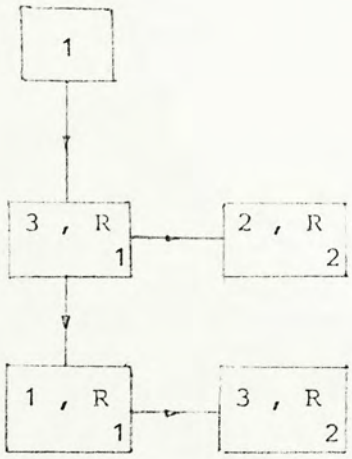
$1+3R_1+2R_2+R_1R_2$



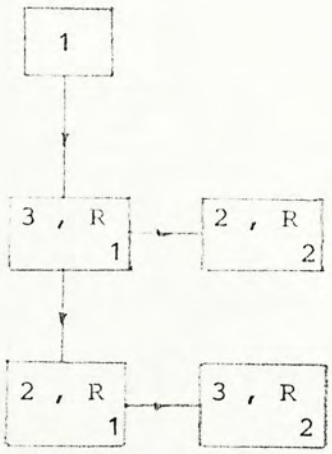
$1+3R_1+2R_2+2R_1R_2$



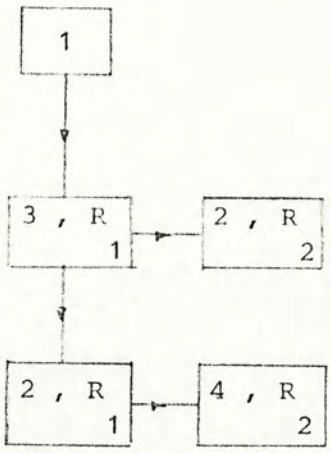
$1+3R_1+2R_2+R_1+2R_1R_2$



$1+3R_1+2R_2+R_1+3R_1R_2$



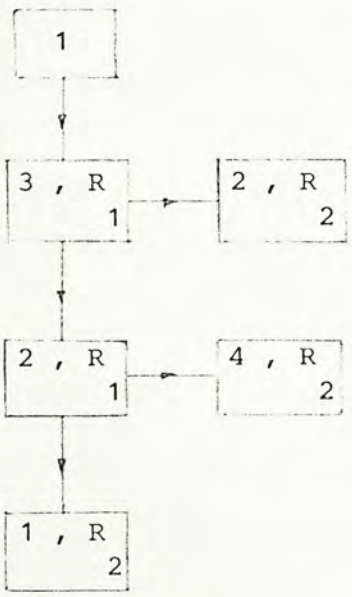
$1+3R_1+2R_2+2R_1+3R_1R_2$



$1+3R_1+2R_2+2R_1+3R_1R_2$

Since both the first two loop weights are $R_1 R_2$, a new node is created in the first step and the coefficient of the node is changed in the second step. In order to produce a unique representation of symbolic expression, a new node representing the third loop weight, R_1^2 is created to replace the position of the symbolic term $R_1 R_2$. The replacement can be easily performed by changing the pointers. The last three terms are then added by changing the corresponding coefficients.

Since there is only one third order loop, the expansion of the determinant is generated after adding the third order loop weight, the final result is given in following tree structure:



$$1+3R_1+2R_2+2R_1^2+4R_1R_2+R_1^2R_2$$

The expansion of the determinant becomes:

$$\Delta = 1 + 3R_1 + 2R_2 + 2R_1^2 + 4R_1R_2 + R_1^2R_2 \qquad (4.22)$$

By using the tree structure symbolic manipulation, a lot of the repeated symbolic terms are cancelled, and we can easily obtain Eq.(4.22) from Eq.(4.21).

Use of tertary tree for tree structure symbolic manipulation

We may use tertary tree instead of binary tree to represent a symbolic expression. Two branches ET and OT are used instead of VT to represent the nodes of next higher level, such that all nodes with even symbol numbers are connected by ET and all the nodes with odd symbol numbers are connected by OT. By implementing the tertary tree symbolic manipulation method, we find that the computer time is reduced by about 5% as comparing to the case of binary tree because the searching time is reduced. On the contrary, the required computer storage is increased by 25% because 5 subfields are used to represent a node. In fact, we sacrify the computer storage to save the computer time.

Conclusion

The new method is an efficient symbolic manipulation method, which can be applied to cancel the repeated symbolic terms in generation of network functions. The method is applied together with some other algorithms to evaluate a symbolic determinant such as Gaussmann Algebra or Signal Flow Graph method. The use of tree structure is particularly easy to be handled with digital computer because operations of tree structure can be easily implemented by using recursive programming technique. Therefore arithematic operations on symbolic expression can be easily performed by applying this symbolic manipulation method.

Chapter 5. Comparsion of Results

In order to investage the efficiency of the new method. We compare it with two other methods: namely, parameter extraction method and the method of Symbolic Network Analysis by means of Computing Numerical Determinants whcih will be given in the appendices. Programs implementing the three methods have been written and named as follows:

1. SERBIT : Symbolic Expression Representation by BInary Tree data structure,
2. SNACOND : Symbolic Network Analysis by means of COmputing Numerical Determinants, and
3. PEM : Parameter Extraction Method.

Several examples which are based on ladder network, are analyzed on the IBM 3031 computer system. In each example, the logarithm of computer time versus number of circuit nodes is plotted. From the slopes of such graphs, we can find out the relation between the rate of increase of computer time and the dimension of the corresponding circuit.

Example 5.1 : Generation of Fully Symbolic Network Functions.

Consider the ladder network shown in Fig. 5.1: all the circuit elements are symbolic parameters and all of them only appear once. In this example, we are going to investigate the case of generation of symbols fully involed network functions.

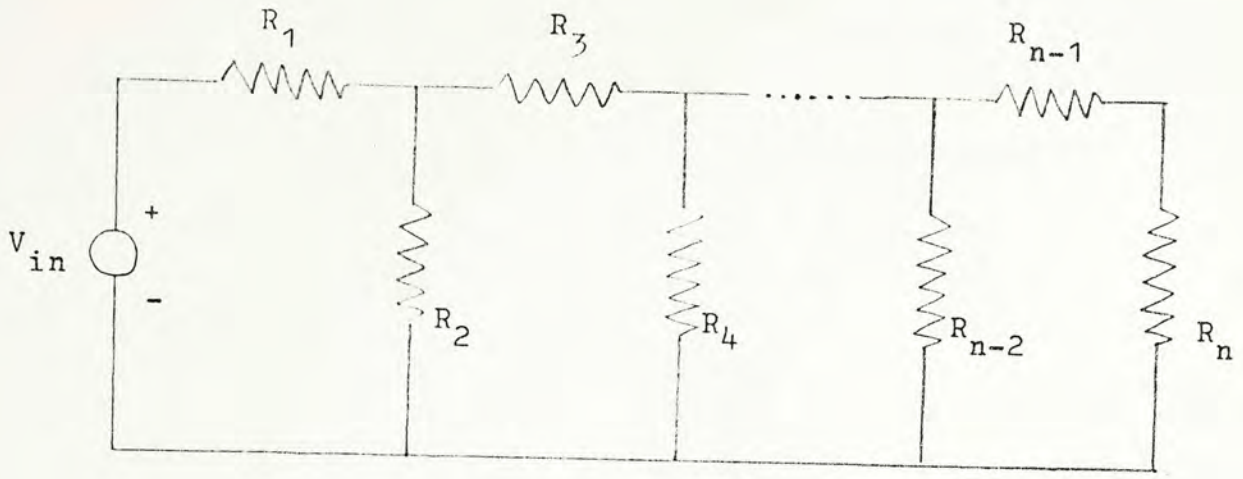


Fig. 5.1 : Ladder network with distinct symbols for all circuit elements.

Fig. 5.2 plots the logarithm of computer times of different methods as functions of number of circuit nodes. From Fig. 5.2, we find that the logarithm of computer times of the three methods increase linearly when the number of nodes of network is larger than 4. The slopes of all the curves are nearly the same; this implies that the rates of increase of computer times for all the methods mentioned are nearly the same. The value of the slopes is approximately 0.610, and the rate of increase of computer time for each additional circuit node is 4.074 ($10^{0.610}$).

Example 5.2 : Generation of Partially Symbolic Network Functions with Fixed Number of Symbolic Parameters.

Consider a ladder network shown in Fig. 5.3 [5], the number of symbols is fixed and we are going to investigate the relationship between the computer time and the size of network.

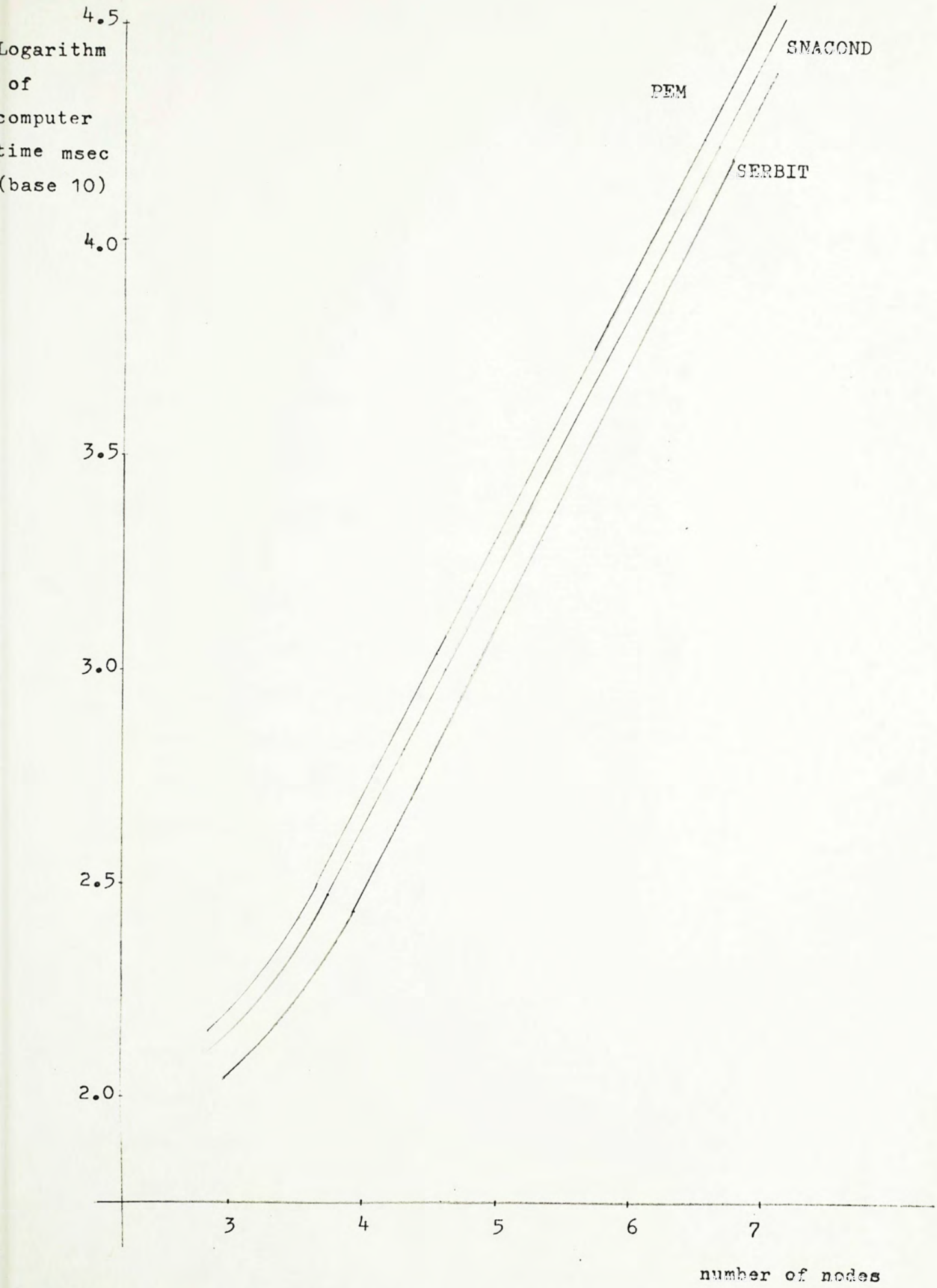


Fig. 5.2 : Logarithm of computer time versus number of nodes of the network shown in Fig. 5.1.

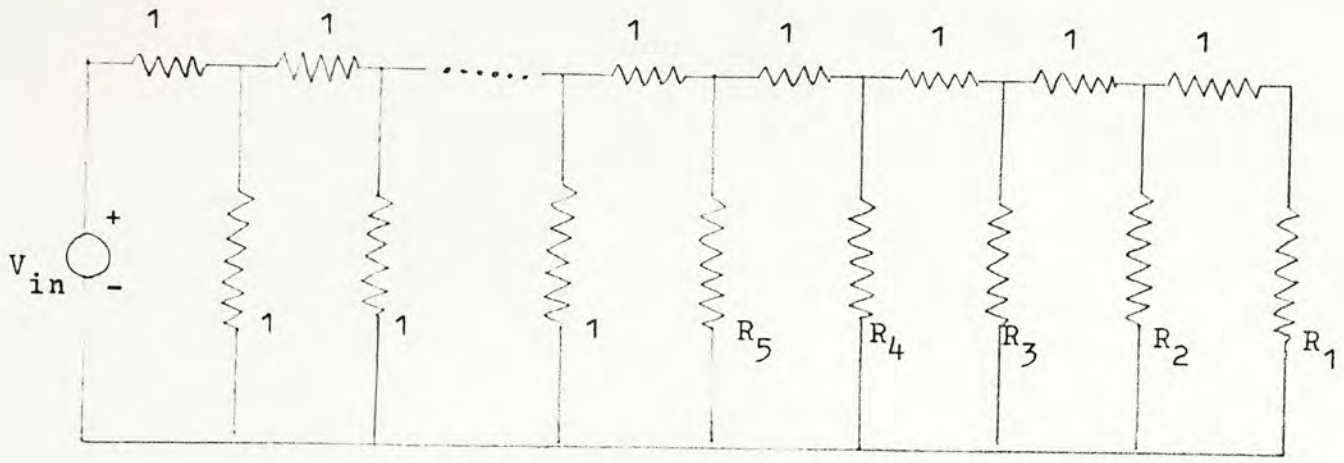


Fig. 5.3 : Ladder network with fixed number of symbolic parameters (R_1, R_2, R_3, R_4 and R_5 are symbols, all other are one ohm resistors).

Fig. 5.4 shows the computer times in logarithm obtained by different methods versus number of circuit nodes studied. (For SERBIT, we use extraction of symbolic parameters in cofactor form instead of using Gaussmann Algebra to expand a partially symbolic determinant. For other examples, Gaussmann Algebra is used to expand symbolic determinants.) Fig. 5.4 shows that the rates of increase of computer times become slower for larger circuits. The networks that can be handled in this example is much larger than the networks of fully symbolic involed case. Moreover, only a little time is used and this is true even for large networks. It means that the rate of increase of computer times depends on the number of symbols included rather than the dimension of determinant studied. Therefore the new method SERBIT has the same potential as PEM and SNACOND to handle large circuits with a few symbols.

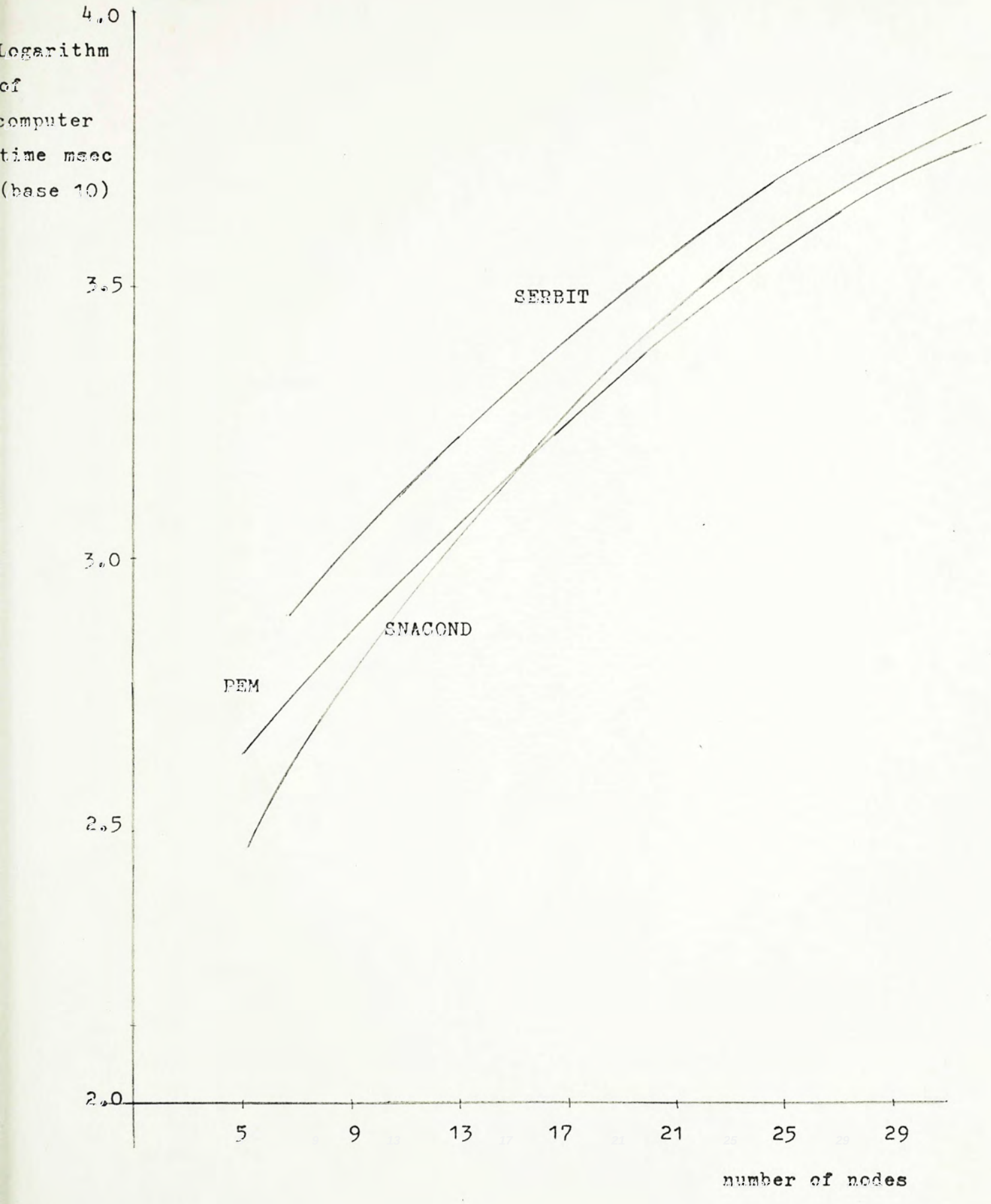


Fig. 5.4 : Logarithm of computer time versus number of circuit nodes of the network shown in Fig. 5.3.

Example 5.3 : Generation of Partially Symbolic Network Function with Distinct Symbols.

In example 5.2, the number of symbols is fixed. In this example, the number of symbolic elements increases as number of circuit nodes. We are going to investigate the relation between the computer time and the number of symbols. Consider the ladder network in Fig. 5.5, the number of distinct symbols is equal to the number of nodes of the network.

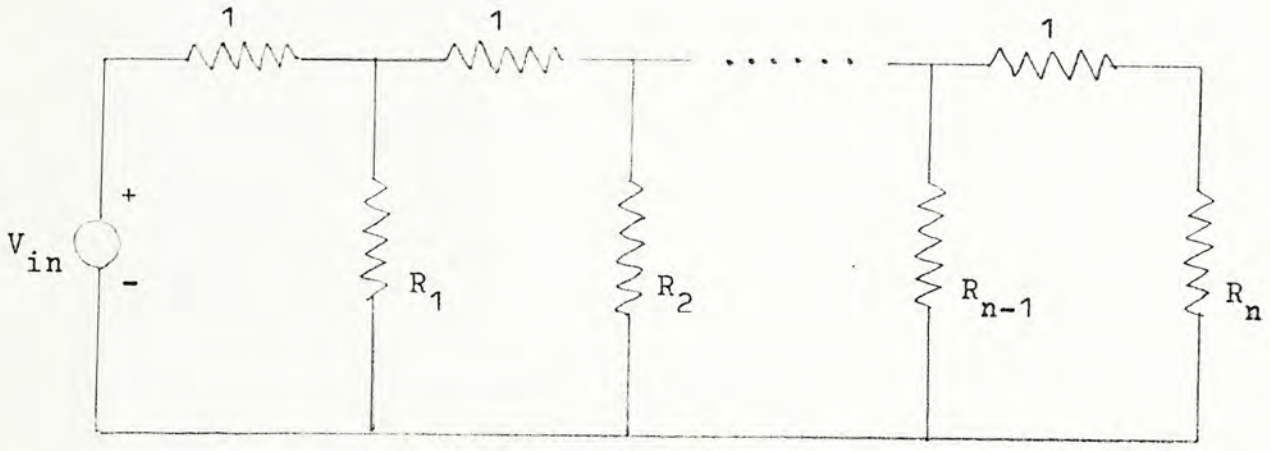


Fig. 5.5 : Ladder network with distinct symbolic circuit elements. (The number of symbolic circuit elements n is equal to the number of nodes.)

Fig. 5.6 plots the computer times used in the three methods versus the number of nodes (number of symbols). In Fig. 5.6, we find that SERBIT is better than both SNACOND and PEM. The computer times in logarithm increase linearly when the number of nodes is larger than 6. The slopes of the graphs SERBIT, SNACOND and PEM are 0.334, 0.357 and 0.408, and the computer times increase with rates of 2.158, 2.275 and 2.559 respectively. Therefore we say that SERBIT is more efficient than SNACOND and PEM to handle a circuit with more symbolic elements.

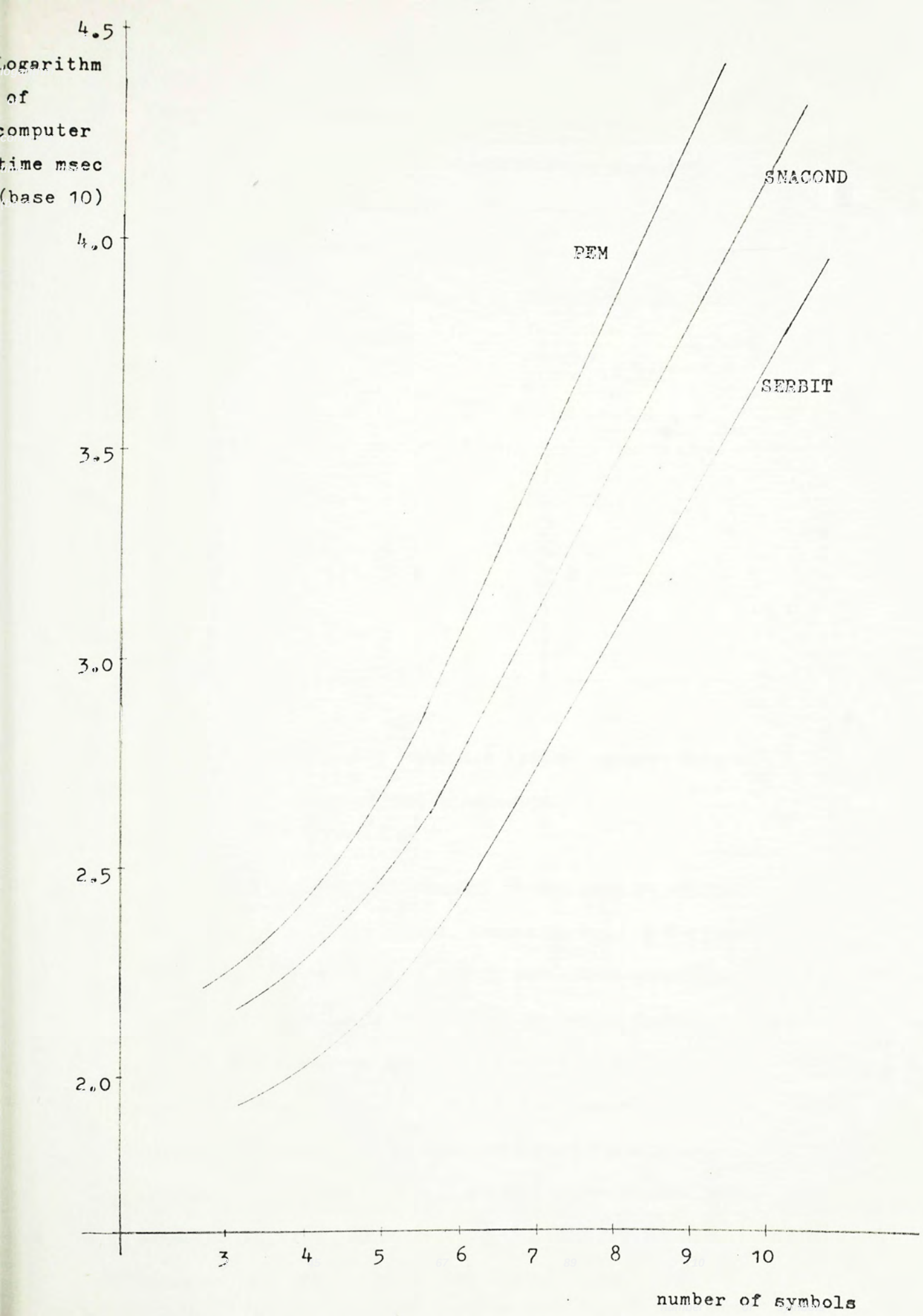


Fig. 5.6 : Logarithm of computer time versus number of distinct symbols.

Example 5.4 : Generation of Partially Symbolic Network Functions with All Symbolic Elements being Repeated.

In the previous example, all symbolic elements are distinct. Now we want to investiage the case of repeated circuit elements. Consider the ladder network shown in Fig. 5.7, all the symbolic circuit elements in the circuit are the same.

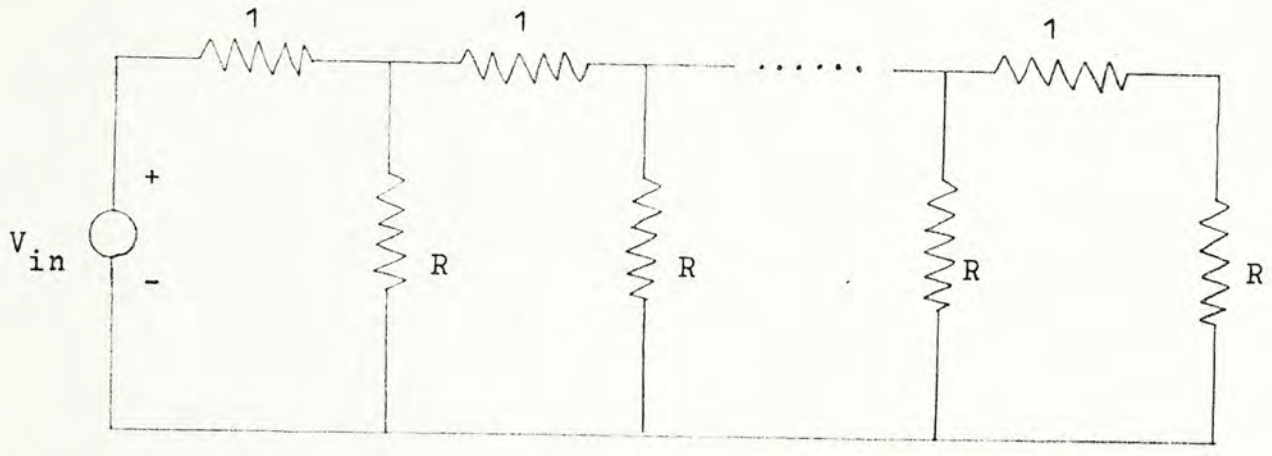


Fig. 5.7 : Partially symbolic ladder network with all the symbolic parameter being repeated.

Fig. 5.8 shows the computer times used in the three methods versus the number of circuit nodes. Comparing Fig. 5.6 with Fig 5.8, we note that the computer time of PEM in both cases remain nearly unchanged. It means that the computer time for parameter extraction method depends only on the number of symbolic elements in the circuit, but takes little time for their repetition. But in the case of SERBIT and SNACOND, the required computer time is much less than the computer time necessary for the case of distinct symbolic circuit elements. The slopes of curves for SERBIT, SNACOND and PEM are 0.243, 0.357 and 0.431, and the rate of increase of computer time are 1.750, 2.275 and 2.698 respectively.

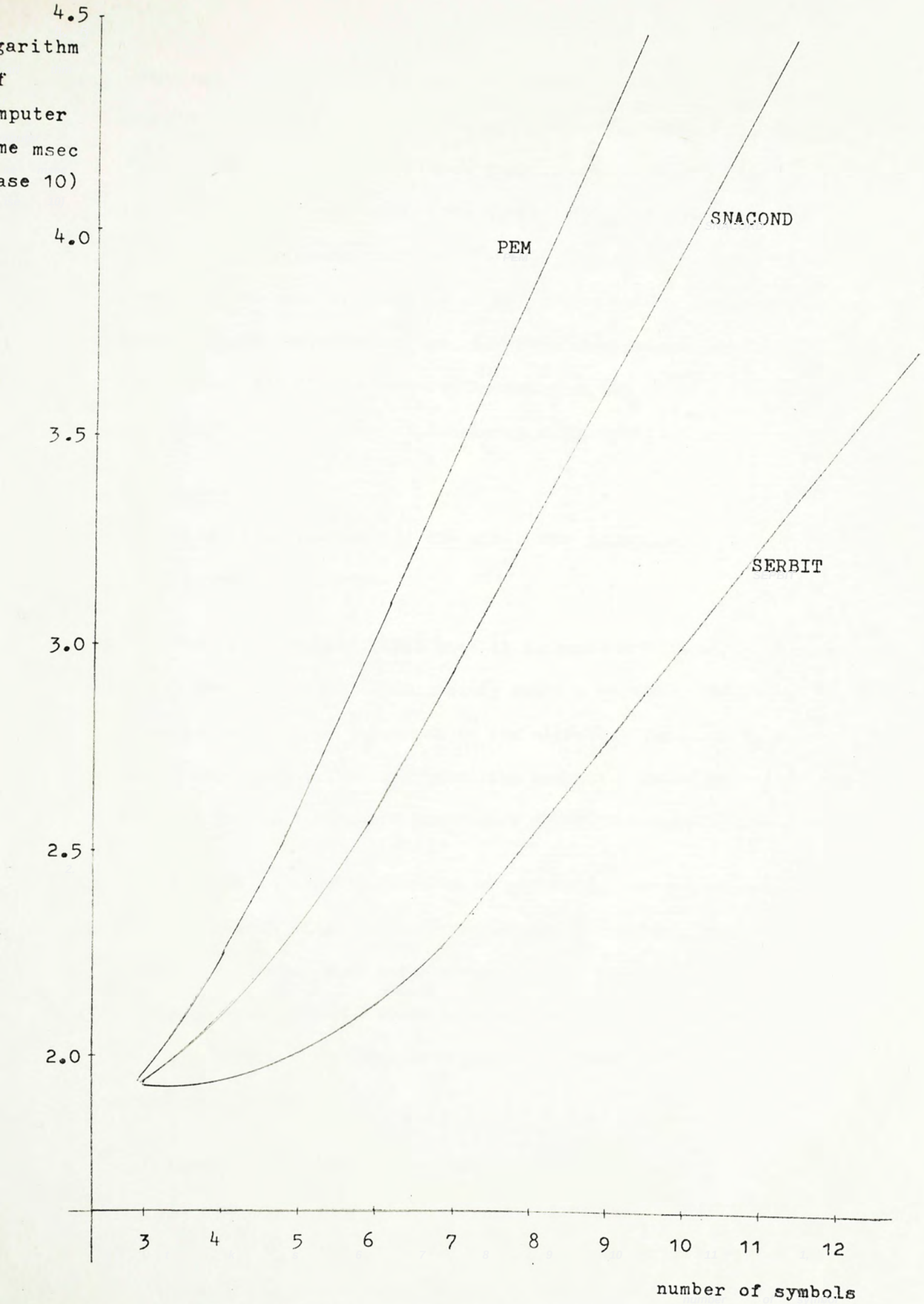


Fig. 5.8 : Logarithm of computer time versus number of repeated symbols.

Comparing these values to the corresponding values in Example 5.3, we see that the slope for PEM becomes larger; this means that computer time of PEM ,becomes increase more rapidly when handling the problem of repeated symbolic elements. The slope of SNACOND remains the same as in Fig. 5.6, it means that even though the computer time of SNACOND is reduced, the rate of increase of computer time has no improvement. Only SERBIT shows improvement on the rate of increase of computer time therefore, SERBIT is more efficient than the other two methods for generating symbolic network functions with repeated circuit elements.

Conclusion

From the results of the above four examples, we can draw the the following conclusion.

PEM is quite efficient when it is applied to a partially symbolic case. The computing time mainly depends on the number of symbolic elements and their locations in the circuit, but takes no account of their repetition, because all the symbolic parameters repeated or distinct in the indefinite admittance matrix are extracted one by one.

SNACOND is another approach of parameter extraction method, it is more efficient than the existing parameter extraction method in most cases. It shows particular advantage for generating symbolic network functions with repeated symbolic circuit elements, because the number of possible numerical determinants formed is less than the case of repeated symbolic parameters.

SERBIT is an efficient symbolic manipulation method. With its aid, we can evaluate a symbolic determinant directly. There are two ways to

expand an symbolic determinant: Gaussmann Algebra or extraction of symbolic terms in cofactor form. In general, we apply Gaussmann Algebra to expand sparse symbolic determinants. When the number of symbolic entries is small as comparing to the number of numerical entries, it is more efficient to use extraction of symbolic terms in the cofactor form. With the aid of such an efficient symbolic manipulation, the method of direct expansion of symbolic determinant is better than the other two methods for generating symbolic network functions.

References

1. P.M. Lin, "A Survey of Applications of Symbolic Network Functions", IEEE Transactions on Circuit Theory, Vol. CT-20, No.6, Nov., 1973.

2. L.O. Chua and O.M. Lin, Computer Aided Analysis of Electronic Circuit : Algorithms and Computational Techniques, Prentice-Hall, 1975.

3. Mason and Zimmermann, Electronic Circuits, Signals and Systems, Wiley, 1960.

4. S.D. Shieu and S.P. Chau, "Topological Formulation of Symbolic Network Functions and Senitivity Analysis of Active Networks", IEEE Transactions on Circuits and Systems, Vol. CAS-21, No.1, Jan., 1974.

5. G.E. Alderson and P.M. Lin, "Computer Generation fo Symbolic Network Functions - A New Theory and Implementation", IEEE Transactions on Circuit Theory Vol. CT-20, No.1, Jan., 1973.

6. P. Sannuti and N.N. Puri, "Symoblic Network Analysis - An Algebra Formulation", IEEE Transactions on Circuits and Systems, Vol. CAS-27, No.8, Aug., 1980.

7. Y.K. Tse, "Symbolic Network Function Generation via Discrete Fourier Transform", Fourth year project report of Department of Electronics, The Chinese University of Hong Kong, 1982.

8. C.F. Chen, Y.T. Tsay and R.E. Yates, "An Unified Approach to Deadbeat Systems Design", Computers and Electrical Engineering, Vol.7, 1980.

9. R.V. Patel, "On the Computation of Numerators of Transfer Functions of Linear Systems", IEEE Transactions on Automatic Control, Aug., 1973.

10. William W. Happ, "Flowgraph Techniques for Closed System", IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-2, No.3, May, 1966.

11. S.J. Mason, "Feedback Theory - Some Properties of Signal Flow Graph", Proceedings IRE Vol.41, 1953.

12. C.R. Wylie, Advanced Engineering Mathematics, third edition, Mcgraw-Hill.

13. C.F. Chen and M.B. Ahmad, "Symbolic Evaluating Transfer Functions via Gaussmann Algebra", Internation Journal of Control, 1984.

14. T. Roska, "Generating Network Functions by Digital Computers Using Prime Numbers", Summer School on Circuit Theory, Short Contr., Vol.1, pp.210-217, Prague, 1968.

15. J. P. Tremblay, An introduction to data structure with applications, McGraw-Hill, 1976.

16. A. M. Tenenbaum, Data Structure using PASCAL, Prentice-Hall Sofeware series.

17. D.H. Owens, Feedback and Multivariable System, Institution of Electrical Engineers, 1978. pg.160.

18. Van Valkenburg, Network Analysis, Prentice-Hall.

Appendices

A. Arithematic operations on tree structure

Operations on tree structures mainly depends on recursive programming technique. Arithematics on tree structure are outlined as follows:

1. Addition (or Subtraction)

Subtraction is the same as addition except the coefficients are subtracted instead of being added.

The operation is to add two trees, and the final result is given in the second tree. This subroutine is a recursive procedure.

```
PROCEDURE ADD(TR1,TR2);
IF TR1 <> NIL
THEN IF (symbol number of TR1) > (symbol number of TR2)
     THEN IF TR2↑.HT = NIL  THEN TR2↑.HT = TR1  ELSE ADD(TR1,TR2↑.HT)
     ELSE IF (symbol number of TR1) < (symbol number of TR2)
          THEN insert TR1 into TR2 by changing values of pointers
          ELSE /* symbolic term is matched */
               add the coefficients of TR1 to TR2
               IF TR2↑.HT = NIL
               THEN TR2↑.HT = TR1↑.HT  ELSE ADD(TR1↑.HT,TR2↑.HT);
               IF TR2↑.VT = NIL
               THEN TR2↑.VT = TR1↑.VT  ELSE ADD(TR1↑.VT,TR2↑.VT)
END.
```

2. Multiplication

Two trees are multiplied together and the result is given in the third tree. Before defining the procedure of multiplication of two trees, we first define a procedure which multiplies a node to a tree. The two procedures are also recursive.

The procedure TRANOD(TR,NOD) multiplies the symbolic term of the node, NOD to the tree, TR, and the result is given in the result tree RTR.

```
PROCEDURE TRANOD(TR,NOD);
IF TR <> NIL
THEN  new term = symbolic term of NOD * symbolic term of current node
                 of TR;
      insert the new term into RTR;
      /* multiply NOD to the branches of TR     */
      TRANOD(TR↑.HT,NOD);
      TRANOD(TR↑.VT,NOD)
END.
```

The following procedure multiplies two trees together. The result is given in the third tree TR3. It first calls TRANOD to multiply the current node of TR2 to the whole tree of TR1, then calls itself again to multiply the branches of TR2 to TR1.

```
PROCEDURE MULT(TR1,TR2)

IF TR2 <> NIL

THEN new tree = TRANOD(TR1, current node of TR2);

     TR3 = ADD(new tree, TR3);

     MULT(TR1,TR2↑.HT);

     MULT(TR1,TR2↑.VT)

END.
```

B. Symbolic Network Analysis by means of Computing Numerical Determinants

With the aid of the closed system technique described in chapter 2, we can simply obtain a symbolic network function by expanding a symbolic determinat. The method to be presented is a numerical approach method to expand a symbolic determinant. It is another approach of parameter extraction method for generation of symbolic network functions, but there is a great difference between this method and the existing parameter extraction method. The existing parameter extracting method extracts the symbolic variables from an indefinite admittance matrix by using the special properties of the indefinite admittance matrix. While this method extracts the symbolic variables from a symbolic determinant by using some general properties of determinant. It extracts the symbolic variables in each row or column from the symbolic determinant until no symbol exists in the determinant and numerical determinant remains. Then we can find the coefficient of an extracted symbolic term by summing up the values of all the possible numerical determinants associated with that symbolic term.

The New Method

We shall use the follow two basic properties of determinant to derive an expansion formula for symbolic determinant:

Theorem B.1

Consider a determinant of dimension n, if the elements in one column (or row) are expressed as binomials, the determinant can be written as sum of two determinants as follow:

$$\begin{vmatrix} a_{11} & \cdots & (p_{1j}+q_{1j}) & \cdots & a_{1n} \\ a_{21} & \cdots & (p_{2j}+q_{2j}) & \cdots & a_{2n} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ a_{n1} & \cdots & (p_{nj}+q_{nj}) & \cdots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & \cdots & p_{1j} & \cdots & a_{1n} \\ a_{21} & \cdots & p_{2j} & \cdots & a_{2n} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ a_{n1} & \cdots & p_{nj} & \cdots & a_{nn} \end{vmatrix} + \begin{vmatrix} a_{11} & \cdots & q_{1j} & \cdots & a_{1n} \\ a_{21} & \cdots & q_{2j} & \cdots & a_{2n} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ a_{n1} & \cdots & q_{nj} & \cdots & a_{nn} \end{vmatrix} \qquad (B.1)$$

Theorem B.2

If all the elements in a column (or row) of a determinant are multiple of X, then X can be extracted from the determinant.

$$\begin{vmatrix} a_{11} & \cdots & Xa_{1j} & \cdots & a_{1n} \\ a_{21} & \cdots & Xa_{2j} & \cdots & a_{2n} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ a_{n1} & \cdots & Xa_{nj} & \cdots & a_{nn} \end{vmatrix} = X \begin{vmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2j} & \cdots & a_{2n} \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & \cdot & & \cdot \\ a_{n1} & \cdots & a_{nj} & \cdots & a_{nn} \end{vmatrix} \qquad (B.2)$$

The above two properties of determinant can be easily proved by expanding the determinant in cofactor form at j-th column [12].

It is well known that the standard methods of network analysis, such as nodal or loop analysis, express network function as fraction of two determinants. By applying closed system technique, symbolic network function can be generated by expanding one symbolic determinant. Thus

generation of symbolic network function is no more than expansion of symbolic determinant. In general, each entry of the determinant will be a linear combination of distinct symbols. Consider a nxn symbolic determinant A with r different symbols ($X_1, X_2, \ldots, X_r$), and a_{ij} is the entry at i-th row and j-th column of A. In terms of the r distinct symbols, a_{ij} can be expressed as

$$a_{ij} = c_{ij,1} X_1 + c_{ij,2} X_2 + \ldots\ldots + c_{ij,r} X_r \qquad (B.3)$$

where $c_{ij,k}$ is the coefficient of symbol X_k (k=1, 2, ..., r) at i-th row and j-th column, and $c_{ij,k}$ =0 means that the symbol X_k does not exist in the entry a_{ij}.

For the case of partially symbolic determinants, the constant term will exist in each entry of the determinant. We can treat the constant terms as the coefficients of a symbol X_o, which is equal to unity and is no need to be extracted. Thus we can concentrate to the case of fully symbolic determinants.

To expand the symbolic determinant A, we first apply theorem B.1 to A at the first column and regard a_{i1} is divided into two parts

$c_{i1,1} X_1$ and $c_{i2,2} X_2 + \ldots\ldots + c_{i1,r} X_r$ yeilds

$$A = \begin{vmatrix} c_{11,1} X_1 & a_{12} & \cdots & a_{1n} \\ c_{21,1} X_1 & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ c_{n1,1} X_1 & a_{n2} & \cdots & a_{nn} \end{vmatrix} + \begin{vmatrix} (c_{11,2} X_2 + \cdots + c_{11,r} X_r) & a_{12} & \cdots & a_{1n} \\ (c_{21,2} X_2 + \cdots + c_{21,r} X_r) & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ (c_{n1,2} X_2 + \cdots + c_{n1,r} X_r) & a_{n2} & \cdots & a_{nn} \end{vmatrix}$$

or, in vector notation,

$$A = \left| \underline{c}_{1,1} X_1 \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right| + \left| (\underline{c}_{1,2} X_2 + \cdots + \underline{c}_{1,r} X_r) \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right| \tag{B.4}$$

where

$$\underline{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \cdot \\ \cdot \\ \cdot \\ a_{nj} \end{bmatrix}, \text{ and } \underline{c}_{j,k} = \begin{bmatrix} c_{1j,k} \\ c_{2j,k} \\ \cdot \\ \cdot \\ \cdot \\ c_{nj,k} \end{bmatrix} \tag{B.5}$$

$j = 1\ 2, \ldots\ldots, n \quad ; \quad k = 1, 2, \ldots\ldots, r$

Then applying theorem B.2 to the first determinant of Eq.(B.4) to extract the symbol X_1 out of the determinant, we have

$$A = X_1 \left| \underline{c}_{1,1} \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right| + \left| (\underline{c}_{1,2} X_2 + \cdots + \underline{c}_{1,r} X_r) \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right| \tag{B.6}$$

Repeatedly applying theorem B.1 and theorem B.2 to A in a similar fashion, we can extracts all the symbols in the first column:

$$A = \left| \underline{a}_1 \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right|$$

$$= X_1 \left| \underline{c}_{1,1} \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right| + X_2 \left| \underline{c}_{1,2} \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right| + \cdots + X_r \left| \underline{c}_{1,r} \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right|$$

$$= \sum_{k_1=1}^{r} X_{k_1} \left| \underline{c}_{1,k_1} \quad \underline{a}_2 \quad \cdots \quad \underline{a}_n \right| \qquad \text{(B.7)}$$

If we repeat the same process to extract symbols from all the other columns, finally we get the expression:

$$A = \sum_{k_1=1}^{r} \sum_{k_2=1}^{r} \cdots \sum_{k_n=1}^{r} X_{k_1} X_{k_2} \cdots X_{k_n} \left| \underline{c}_{1,k_1} \quad \underline{c}_{2,k_2} \quad \cdots \quad \underline{c}_{n,k_n} \right| \qquad \text{(B.8)}$$

Grouping all the terms with same symbolic product $X_{u_1} X_{u_2} \cdots X_{u_n}$ together ($u_j = 1, 2, \ldots, r$; $j = 1, 2, \ldots, n$), we can rewrite Eq.(B.8) as

$$A = \sum_{\substack{\text{all combination} \\ \text{of } u_1\ u_2\ \cdots\ u_n}} X_{u_1} X_{u_2} \cdots X_{u_n} \sum_{\substack{\text{all distinct} \\ \text{permutation of} \\ u_1\ u_2\ \cdots\ u_n}} \left| \underline{c}_{1,u_1} \quad \underline{c}_{2,u_2} \quad \cdots \quad \underline{c}_{n,u_n} \right| \qquad \text{(B.9)}$$

where the first summation sign sums up all the distinct symbolic terms, and the second summation sign sums up all the values of numerical determinants to form the coefficient of one symbolic term.

Remarks

According to Eq.(B.9), to find the coefficient of a symbolic term, the number of numerical determinants to be evaluated is not the same; for example, we can find the coefficient of the symbolic term X_1^n by evaluating only one numerical determinant, while we need to evaluate n! numerical determinants in order to find the coefficient of the symbolic term $X_1 X_2 \ldots X_n$. In general, the number of numerical determinants to be evaluated for terms with symbols of higher power will be less than for terms with nearly distinct symbols. Thus this method is particularly efficient for generating symbolic network functions with a large number of repeated symbolic elements.

At first sight, the proposed approach might not appear to be very useful because there are r^n possible numerical determinants to be evaluated as can be seen from Eq.(B.8) and the number of determinants to be evaluated will grow rapidly as r and n increase. In fact, the number of valid terms will be much reduced owing to the following reasons:

1. Valid symbolic terms are limited by the number of occurences of symbolic circuit elements in the network.

   Usually the number of occurences of a symbolic circuit element is much less than the dimension of the determinant. For example, if n=10 and if X_1 occurs only three times in the circuit, then all the

terms with a power of X_1 greater than three are invalid.

2. Sparse determinants are often encountered in network analysis.

Usually the determinants formed by say, nodal or loop analysis are sparse; each symbol will only occur at a few locations in the determinants. There are many determinants in Eq.(B.9) which will have a vanishing row or column and can therefore be neglected.

3. All valid symbolic term are products of n symbols.

Since each entry of the determinant is a linearly combination of symbols, and a valid term is formed as product of entries at different row and different column, there must be n symbols in a valid symbolic term. Thus all the terms whose number of symbols are not n, are invalid.

The above points can be illustrated by the following example.

Example B.1

Consider the network shown in Fig. B.1.

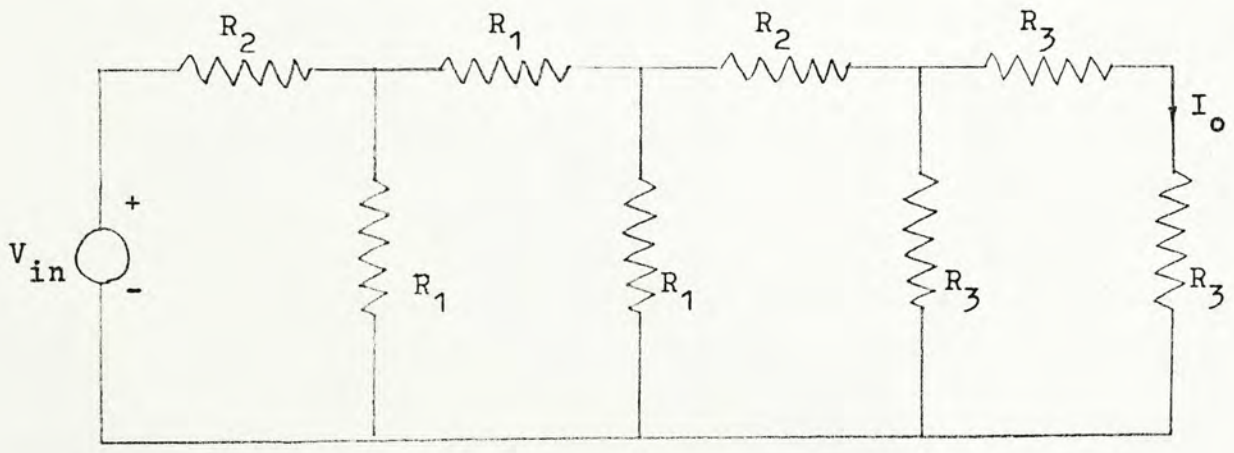


Fig. B.1 : A simple network.

Using loop analysis, we find the network function as follow:

$$T = \frac{I_o}{V_i} = \frac{N}{D} = \frac{\begin{vmatrix} -R_1 & 3R_1 & -R_1 & \\ 0 & -R_1 & R_1+R_2+R_3 & \\ 0 & 0 & -R_3 & \end{vmatrix}}{\begin{vmatrix} R_1+R_2 & -R_1 & 0 & 0 \\ -R_1 & 3R_1 & -R_1 & 0 \\ 0 & -R_1 & R_1+R_2+R_3 & -R_3 \\ 0 & 0 & -R_3 & 3R_3 \end{vmatrix}} \qquad \text{(B.10)}$$

Consider the denominator only,

$$D = \begin{vmatrix} R_1 \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + R_2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + R_3 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 3 \end{bmatrix} \end{vmatrix} \qquad \text{(B.11)}$$

$$D = \begin{vmatrix} R_1 \begin{bmatrix} \underline{c}_{1,1} & \underline{c}_{2,1} & \underline{c}_{3,1} & 0 \end{bmatrix} + R_2 \begin{bmatrix} \underline{c}_{1,2} & 0 & \underline{c}_{3,2} & 0 \end{bmatrix} + R_3 \begin{bmatrix} 0 & 0 & \underline{c}_{3,3} & \underline{c}_{4,3} \end{bmatrix} \end{vmatrix} \qquad \text{(B.12)}$$

where we see that $\underline{c}_{4,1}$, $\underline{c}_{2,2}$, $\underline{c}_{4,2}$, $\underline{c}_{1,3}$ and $\underline{c}_{2,3}$ are all zero.

Expanding D in accordance with Eq.(B.9) gives

$$D = R_1 R_2 R_3^2 \begin{vmatrix} \underline{c}_{1,2} & \underline{c}_{2,1} & \underline{c}_{3,3} & \underline{c}_{4,3} \end{vmatrix} + R_1 R_2 R_3^2 \begin{vmatrix} \underline{c}_{1,2} & \underline{c}_{2,1} & \underline{c}_{3,2} & \underline{c}_{4,3} \end{vmatrix} +$$

$$R_1^2 R_3^2 \begin{vmatrix} \underline{c}_{1,1} & \underline{c}_{2,1} & \underline{c}_{3,3} & \underline{c}_{4,3} \end{vmatrix} + R_1 R_3^3 \begin{vmatrix} \underline{c}_{1,1} & \underline{c}_{2,1} & \underline{c}_{3,1} & \underline{c}_{4,3} \end{vmatrix} +$$

$$R_1^2 R_2 R_3 \left(\begin{vmatrix} \underline{c}_{1,1} & \underline{c}_{2,1} & \underline{c}_{3,2} & \underline{c}_{4,3} \end{vmatrix} + \begin{vmatrix} \underline{c}_{1,2} & \underline{c}_{2,1} & \underline{c}_{3,1} & \underline{c}_{4,3} \end{vmatrix} \right)$$

(B.13)

After evaluating all the numerical determinants, we have

$$D = 6 R_1^2 R_2 R_3 + 9 R_1 R_2^2 R_3 + 4 R_1^2 R_3^2 + 3 R_1 R_3^3 + 12 R_1^2 R_2 R_3$$

(B.14)

To expand the determinant in Eq.(B.11), the number of all possible numerical determinants is $r_n^4 = 3^4 = 81$, but in fact only a total of 6 non-zero determinants are need to be evaluated.

Application of the new method to multivariable system analysis

In chapter two, the closed system technique is extended to the multivariable system analysis. It is shown that we can obtain the transfer matrix by evaluating only one determinant and then sorting all the terms according to the existance of the controlled source symbols. However we need to do some redundant work to evaluate the terms with more than one controlled source symbols because all such terms are not related to the result of the transfer matrix. By using the method described in this section, we can evaluate the symbolic determinant by finding the coefficient of each term seperately.

The new method first extracts all the symbols, then finds the coefficient of each symbolic term by computing some numerical determinants. The numerical determinants are formed by combining all the possible columns (or rows) of different symbols according to the symbolic term and the coefficients are found one by one seperately. Therefore by using this method to evaluate the symbolic determinant of multivariable system, we can only find the coefficients of the symbolic terms related to the transfer matrix and simply eliminate all the redundant work by not finding the coefficients of the unnecessary terms. The use of the new method to analyze a multivariable system is shown in the following example.

Example B.2

Example 2.2 is analyzed more detailed here. The system equation is described as follow:

$$\begin{bmatrix} s+2 & -1 & -1 \\ -1 & s+1+k & -k \\ -1 & -k & s+1+k \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \qquad \text{(B.15)}$$

If the desired outputs are X_1 and X_2, we change the inputs into controlled sources related to the outputs as follows:

$$\begin{aligned} U_1 &= -(P_{11} X_1 + P_{12} X_2) \\ U_2 &= -(P_{21} X_1 + P_{22} X_2) \end{aligned} \qquad \text{(B.16)}$$

Substituting Eq.(B.16) into Eq.(B.15) yeilds

$$\begin{bmatrix} s+2 & -1 & -1 \\ -1 & s+1+k & -k \\ -1 & -k & s+1+k \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -(P_{11} X_1 + P_{12} X_2) \\ -(P_{21} X_1 + P_{22} X_2) \end{bmatrix}$$

$$= \begin{bmatrix} -P_{11} & -P_{12} \\ 0 & 0 \\ -P_{21} & -P_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \qquad \text{(B.17)}$$

Regrouping the symbols X_1 and X_2 gives

$$\begin{bmatrix} P_{11}+s+2 & P_{12}-1 & -1 \\ -1 & s+1+k & -k \\ P_{21}-1 & P_{22}-k & s+1+k \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = 0 \qquad \text{(B.18)}$$

In order to obtain a more systematic procedure to evaluate the system determinant, we first assign each symbol with a symbol number and name the symbols as follows:

Z_o = constant term,

$Z_1 = P_{11}$, $Z_2 = P_{12}$,

$Z_3 = P_{21}$, $Z_4 = P_{22}$,

$Z_5 = s$, $Z_6 = k$ (B.19)

After extracting all the symbols, we have

$$\Delta_{cs} = \left| Z_o \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} + Z_5 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + Z_6 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} + Z_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right.$$

$$\left. + Z_2 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + Z_3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} + Z_4 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \right| \qquad \text{(B.20)}$$

or in vector notation,

$$\Delta_{cs} = \left| Z_o \begin{bmatrix} \underline{c}_{1,o} & \underline{c}_{2,o} & \underline{c}_{3,o} \end{bmatrix} + Z_5 \begin{bmatrix} \underline{c}_{1,5} & \underline{c}_{2,5} & \underline{c}_{3,5} \end{bmatrix} + Z_6 \begin{bmatrix} \underline{0} & \underline{c}_{2,6} & \underline{c}_{3,6} \end{bmatrix} \right.$$

$$+ Z_1 \begin{bmatrix} \underline{c}_{1,1} & \underline{0} & \underline{0} \end{bmatrix} + Z_2 \begin{bmatrix} \underline{0} & \underline{c}_{2,2} & \underline{0} \end{bmatrix} + Z_3 \begin{bmatrix} \underline{c}_{1,3} & \underline{0} & \underline{0} \end{bmatrix}$$

$$\left. + Z_4 \begin{bmatrix} \underline{0} & \underline{c}_{2,4} & \underline{0} \end{bmatrix} \right| \qquad \text{(B.21)}$$

where $\underline{c}_{1,6}$, $\underline{c}_{2,1}$, $\underline{c}_{3,1}$, $\underline{c}_{1,2}$, $\underline{c}_{3,2}$, $\underline{c}_{2,3}$, $\underline{c}_{3,3}$, $\underline{c}_{1,4}$, and $\underline{c}_{3,4}$ are all zero.

After all the symbols are extracted, we can find the coefficients of the symbolic terms by computing numerical determinants formed for each symbolic term and then sorting them according to the existance of controlled source symbols. The denominator of the transfer matrix is equal to the sum of all the terms without any controlled source symbols. The denominator is found as follows:

$$\Delta = \begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,o} & \underline{C}_{3,o} \end{vmatrix} + Z_5 \left(\begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,o} & \underline{C}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,5} & \underline{C}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,5} & \underline{C}_{3,o} \end{vmatrix} \right) + Z_5^2 \left(\begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,5} & \underline{C}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,o} & \underline{C}_{3,5} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,5} & \underline{C}_{3,5} \end{vmatrix} \right) + Z_5^3 \begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,5} & \underline{C}_{3,5} \end{vmatrix} + Z_6 Z_5 \left(\begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,6} & \underline{C}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,6} & \underline{C}_{3,5} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,5} & \underline{C}_{3,6} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,o} & \underline{C}_{3,6} \end{vmatrix} \right) + Z_6 \left(\begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,6} & \underline{C}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,o} & \underline{C}_{3,6} \end{vmatrix} \right) + Z_5^2 Z_6 \left(\begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,5} & \underline{C}_{3,6} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,6} & \underline{C}_{3,5} \end{vmatrix} \right) + Z_6^2 \begin{vmatrix} \underline{C}_{1,o} & \underline{C}_{2,6} & \underline{C}_{3,6} \end{vmatrix} + Z_6^2 Z_5 \begin{vmatrix} \underline{C}_{1,5} & \underline{C}_{2,6} & \underline{C}_{3,6} \end{vmatrix}$$

$$\Delta = (0) + s\,(1+1+1) + s^2\,(1+1+2) + s^3\,(1) + k\,s\,(1+2+2+1) + k\,(0+0) + s^2 k\,(1+1) + k^2\,(0) + k^2 s\,(0)$$

$$= 3s + 4s^2 + s^3 + 6ks + 2s^2 k \qquad \text{(B.22)}$$

N_{11} is equal to the sum of all the symbolic terms with the controlled source symbol $P_{11}(Z_1)$. After extracting Z_1, N_{11} is found as follow:

$$N_{11} = \begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,o} & \underline{C}_{3,o} \end{vmatrix} + Z_5 \left(\begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,5} & \underline{C}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,o} & \underline{C}_{3,5} \end{vmatrix} \right) + Z_5^2 \begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,5} & \underline{C}_{3,5} \end{vmatrix} + Z_6 \left(\begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,6} & \underline{C}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,o} & \underline{C}_{3,6} \end{vmatrix} \right) + Z_6^2 \begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,6} & \underline{C}_{3,6} \end{vmatrix} + Z_5 Z_6 \left(\begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,5} & \underline{C}_{3,6} \end{vmatrix} + \begin{vmatrix} \underline{C}_{1,1} & \underline{C}_{2,6} & \underline{C}_{3,5} \end{vmatrix} \right)$$

$$= (1) + s\,(1+1) + s^2\,(1) + k\,(1+1) + k^2\,(0) + s\,k\,(1+1)$$

$$= 1 + 2s + s^2 + 2k + 2sk \qquad \text{(B.23)}$$

From the above determinants, we find that the first column of all the determinants is $\underline{c}_{1,1}$. For the same output with different inputs, we can simply generate the numerical determinants just by changing the first column. To generate all the possible numerical determinants for the symbolic terms of N_{21}, we can simply change the first column of all the numerical determinants in Eq.(B.22) to $\underline{c}_{1,3}$. Therefore the procedure to generate the possible numerical determinants for N_{21} can be eliminated. N_{21} is found as follows:

$$N_{21} = \begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,o} & \underline{c}_{3,o} \end{vmatrix} + Z_5 \left(\begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,5} & \underline{c}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,o} & \underline{c}_{3,5} \end{vmatrix} \right) +$$

$$Z_5^2 \begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,5} & \underline{c}_{3,5} \end{vmatrix} + Z_6 \left(\begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,6} & \underline{c}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,o} & \underline{c}_{3,6} \end{vmatrix} \right) +$$

$$Z_6^2 \begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,6} & \underline{c}_{3,6} \end{vmatrix} + Z_5 Z_6 \left(\begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,5} & \underline{c}_{3,6} \end{vmatrix} + \begin{vmatrix} \underline{c}_{1,3} & \underline{c}_{2,6} & \underline{c}_{3,5} \end{vmatrix} \right)$$

$$= (1) + s\,(1+0) + s^2\,(0) + k\,(1+1) + k^2\,(0) + s\,k\,(0+0)$$

$$= 1 + s + 2k \qquad \text{(B.24)}$$

N_{12} is equal to sum of all the symbolic terms with the controlled source symbol $P_{12}(Z_2)$. After extracting Z_2, N_{12} is found as follows:

$$N_{12} = \begin{vmatrix} \underline{c}_{1,o} & \underline{c}_{2,2} & \underline{c}_{3,o} \end{vmatrix} + Z_5 \left(\begin{vmatrix} \underline{c}_{1,5} & \underline{c}_{2,2} & \underline{c}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{c}_{1,o} & \underline{c}_{2,2} & \underline{c}_{3,5} \end{vmatrix} \right) +$$

$$Z_5^2 \begin{vmatrix} \underline{c}_{1,5} & \underline{c}_{2,2} & \underline{c}_{3,5} \end{vmatrix} + Z_6 \begin{vmatrix} \underline{c}_{1,o} & \underline{c}_{2,2} & \underline{c}_{3,6} \end{vmatrix} + Z_5 Z_6 \begin{vmatrix} \underline{c}_{1,5} & \underline{c}_{2,2} & \underline{c}_{3,6} \end{vmatrix}$$

$$= (1) + s\,(0 + 1) + s^2\,(0) + k\,(2 + 0)$$

$$= 1 + s + 2k \qquad \text{(B.25)}$$

N_{22} is equal to sum of all the symbolic term with the controlled source symbol $P_{22}(Z_4)$. The numerical determinants can be formed by changing the second column of all the determinants formed for N_{12} to $\underline{c}_{2,4}$. After extracting Z_4, N_{22} is found as follows:

$$N_{22} = \begin{vmatrix} \underline{c}_{1,o} & \underline{c}_{2,4} & \underline{c}_{3,o} \end{vmatrix} + Z_5 \left(\begin{vmatrix} \underline{c}_{1,5} & \underline{c}_{2,4} & \underline{c}_{3,o} \end{vmatrix} + \begin{vmatrix} \underline{c}_{1,o} & \underline{c}_{2,4} & \underline{c}_{3,5} \end{vmatrix} \right) +$$

$$Z_5^2 \begin{vmatrix} \underline{c}_{1,5} & \underline{c}_{2,4} & \underline{c}_{3,5} \end{vmatrix} + Z_6 \begin{vmatrix} \underline{c}_{1,o} & \underline{c}_{2,4} & \underline{c}_{3,6} \end{vmatrix} + Z_5 Z_6 \begin{vmatrix} \underline{c}_{1,5} & \underline{c}_{2,4} & \underline{c}_{3,6} \end{vmatrix}$$

$$= (1) + s\,(0 + 0) + s^2\,(0) + k\,(2) + s\,k\,(1)$$

$$= 1 + 2k + s\,k \qquad \text{(B.26)}$$

Combining all the above result, we can obtain the transfer matrix as follows:

$$\begin{bmatrix} N_{11} & N_{21} \\ N_{12} & N_{22} \end{bmatrix} = \frac{\begin{bmatrix} 1+2s+s^2+2k+2sk & 1+s+sk \\ 1+s+2k & 1+2k+sk \end{bmatrix}}{3s+4s^2+s^3+6sk+2s^2k} \tag{B.27}$$

From the above example, we see that all the unnecessary symbolic terms are no need to be found. For the same output, we can obtain numerical determinants by only changing the corresponding column with different inputs.

Conclusion

The method is similar to the existing parameter extraction method in philosophy. They first extract all the symbols from a symbolic determinant or indefinite admittance matrix, then evaluate the coefficients of symbolic terms by evaluateing some numerical determinants. The new method is particularly advantageous for the case of network with repeated symbolic circuit elements. Also the method is more suitable for evaluating the transfer matrices in multivariable systems analysis.